AntiNex - Deep Neural Networks for Defense Documentation

Release 1.0.0

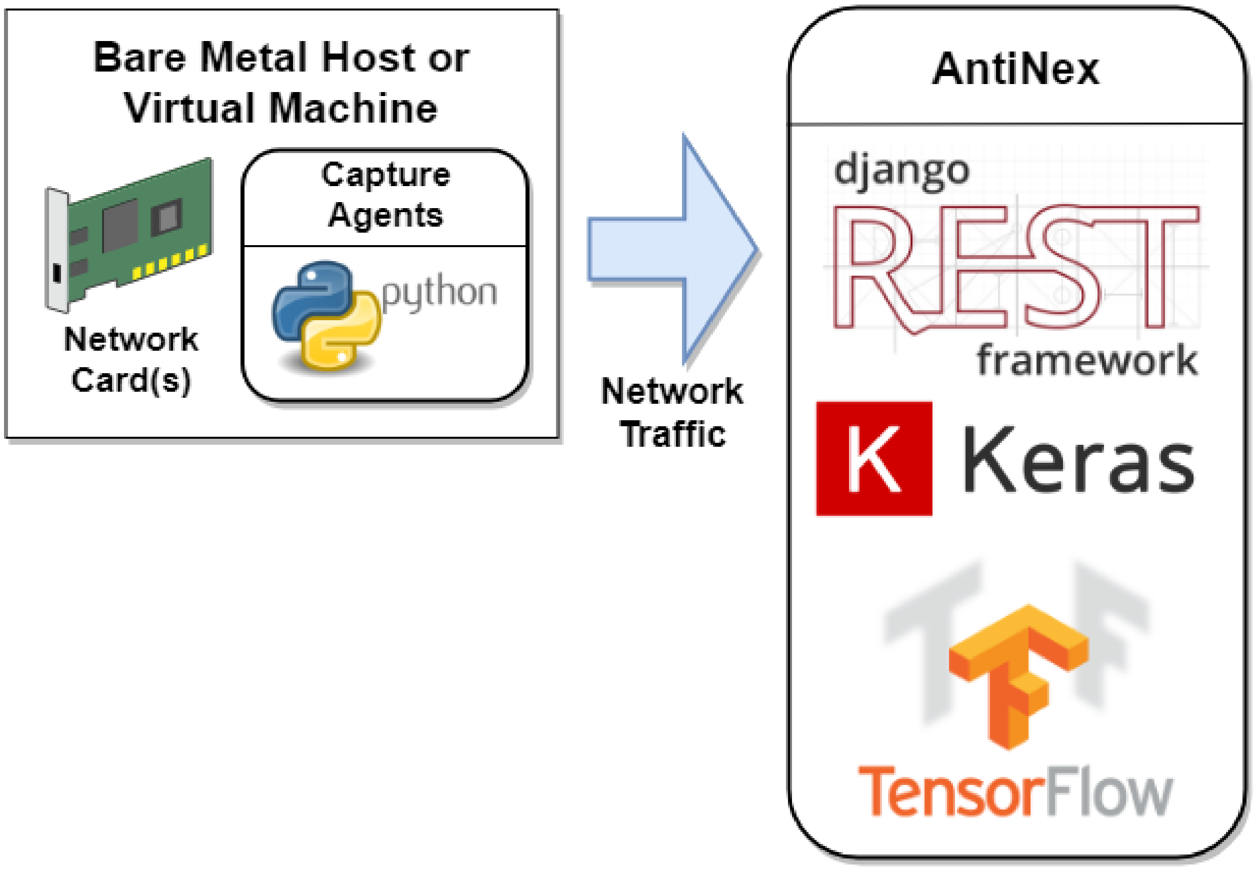
Jay Johnson

Sep 05, 2018

Contents

CHAPTER 1

Deep Neural Networks for Defending Software Systems



CHAPTER 2

What is this?

AntiNex is a free tool for helping anyone defend against software attacks. It helps users train highly accurate Deep Neural Networks (dnn's) from specialized datasets. These datasets are captured network traffic packets in the OSI layers 2, 3, 4 and 5. Once labeled as attack and non-attack records, you can use your dnn's for identifying attack records across the network. With this approach, AntiNex can predict attacks on web applications like: Django, Flask, React and Redux, Vue, and Spring with repeatable accuracies above **99.7%**. By default just one AntiNex Core (core) worker manages 100 pre-trained dnn's in memory for making faster predictions and support for manual retraining as needed based off new datasets.

- AntiNex core accuracy scores
- Jupyter notebook for how it works without any of the AntiNex components as proof of the methodology
- Jupyter notebook for using a pre-trained dnn to make new predictions with AntiNex

AntiNex is a python 3 multi-tenant framework for running a data pipeline for building, training, scoring and refining dnn's. Once trained, dnn's can be loaded into the core for making predictions in near-realtime as the models have already been tuned and pre-trained. The initial focus of AntiNex was to create AI models to defend web applications, but it makes predictions with classification (used for labeling attack vs non-attack records) or regression (like predicting the closing price of a stock) datasets.

CHAPTER 3

Quick Start

3.1 Deploy on OpenShift Container Platform

Deploy AntiNex on Red Hat's OpenShift Container Platform (version 3.9)

3.2 Local Deployment with Docker Compose

If you have docker-compose you can run the following commands to download all the containers and run the full stack locally (the ai-core container is ~2.5 GB so it can take a couple minutes to download):

```
virtualenv -p python3 ~/.venvs/testing
source ~/.venvs/testing/bin/activate
pip install antinex-client
git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt /opt/
↪antinex/api
cd /opt/antinex/api

# start all the containers from the compose.yml file: https://github.com/jay-johnson/
↪train-ai-with-django-swagger-jwt/blob/master/compose.yml
./run-all.sh
Starting all containers with: compose.yml
Creating redis    ... done
Creating jupyter  ... done
Creating pgadmin  ... done
Creating postgres ... done
Creating api      ... done
Creating core     ... done
Creating worker   ... done
Creating pipeline ... done

# check the containers are running
docker ps
```

(continues on next page)

(continued from previous page)

```
CONTAINER ID        IMAGE                           COMMAND                  CREATED           ␣
↪        STATUS               PORTS                    NAMES
cb0d0e8e582e        jayjohnson/ai-core:latest   "/bin/sh -c 'cd /opt..."   33 seconds␣
↪ago       Up 32 seconds                               worker
4b0c44c99472        jayjohnson/ai-core:latest   "/bin/sh -c 'cd /opt..."   33 seconds␣
↪ago       Up 32 seconds                               pipeline
bd3c488036dd        jayjohnson/ai-core:latest   "/bin/sh -c 'cd /opt..."   34 seconds␣
↪ago       Up 33 seconds                               core
a3093e2632b7        jayjohnson/ai-core:latest   "/bin/sh -c 'cd /opt..."   34 seconds␣
↪ago       Up 33 seconds                               api
3839a0af82ec        jayjohnson/pgadmin4:1.0.0   "python ./usr/local/..."   35 seconds␣
↪ago       Up 33 seconds        0.0.0.0:83->5050/tcp     pgadmin
b4ea601f28cd        redis:4.0.5-alpine          "docker-entrypoint.s..."   35 seconds␣
↪ago       Up 33 seconds        0.0.0.0:6379->6379/tcp   redis
c5eb07041509        postgres:10.2-alpine        "docker-entrypoint.s..."   35 seconds␣
↪ago       Up 34 seconds        0.0.0.0:5432->5432/tcp   postgres
9da0440864e0        jayjohnson/ai-core:latest   "/opt/antinex/core/d..."   35 seconds␣
↪ago       Up 34 seconds                                jupyter
```

3.3 Migrate the DB

SSH into the Django container and run the migration:

```
docker exec -it worker bash
cd /opt/antinex/api
./run-migrations.sh
exit
```

3.4 Train the Django Neural Network with 99.8% Accuracy

```
# train a deep neural network with the included antinex-datasets
ai_train_dnn.py -u root -p 123321 -f tests/only-publish-scaler-full-django.json

...
... more logs
...

2018-03-29 20:50:13,306 - ai-client - INFO - started job.id=1 job.status=initial with␣
↪result.id=1 result.status=initial

...
30199    -1.0 -1.000000  -1.000000

[30200 rows x 72 columns]

# Here's how to watch what the containers are doing:
# AntiNex Core:
# docker logs -f core
# AntiNex REST API:
# docker logs -f api
# AntiNex REST API Celery Worker:
# docker logs -f worker
```

3.5 Get the Accuracy, Training and Prediction Results

Return the 30,200 predicted records and accuracy scores (which were 99.826%) from in the database.

```
ai_get_results.py -u root -p 123321 -i 1
2018-03-29 20:52:26,348 - ai-client - INFO - creating client user=root url=http://
↪localhost:8010 result_id=1
2018-03-29 20:52:26,349 - ai-client - INFO - loading request in result_id=1
2018-03-29 20:52:26,360 - ai-client - INFO - log in user=root url=http://
↪localhost:8010/api-token-auth/ ca_file=None cert=None
2018-03-29 20:52:30,876 - ai-client - INFO - accuracy=99.82615894039735 num_
↪results=30200
2018-03-29 20:52:30,876 - ai-client - INFO - done getting result.id=1
```

3.6 Make Predictions with Your New Pre-trained Neural Network

Note: this is using the same HTTP Request JSON dictionary as the initial training, but this time the AntiNex Core will reuse the pre-trained deep neural network for making new predictions.

```
ai_train_dnn.py -u root -p 123321 -f tests/only-publish-scaler-full-django.json

...

30199    -1.0 -1.000000  -1.000000

[30200 rows x 72 columns]
```

3.7 Get the New Prediction Records and Results

```
ai_get_results.py -u root -p 123321 -i 2
```

CHAPTER 4

API Examples

4.1 AntiNex API Examples

Here are a few ways to learn about the AntiNex API.

4.2 AntiNex Python Client within a Jupyter Notebook

Here is how to use the antinex-client for training and using a pre-trained Deep Neural Network to make predictions:

https://github.com/jay-johnson/antinex-core/blob/master/docker/notebooks/AntiNex-Using-Pre-Trained-Deep-Neural-Networks-For-D
ipynb

4.2.1 More Jupyter Links

Login to the Notebook with **admin**:

http://localhost:8888/notebooks/AntiNex-Using-Pre-Trained-Deep-Neural-Networks-For-Defense.ipynb

Presentation Slides running in the Jupyter container (with arrow keys for navigation):

http://localhost:8890/Slides-AntiNex-Using-Pre-Trained-Deep-Neural-Networks-For-Defense.slides.html#/

4.3 Using Curl

4.3.1 Login a User

This will get the JWT token for a user. In this example it is the only user on an initial system **root**:

```
curl -s -X POST \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    -d '{ "username": "root", "password": "123321" }' \
    'http://0.0.0.0:8010/api-token-auth/'

{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MjI0MjM0NzEsImVtYWlsIjoicm9vdEBlbWFpbC5jb20ifQ
↪LBfDvnoG5ilLWgB7lg6EWuR4QkspppF9NHy7oyCmh1s"}
```

If you want to store the token in a simple variable like **token** use:

```
token=$(curl -s -X POST \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    -d '{ "username": "root", "password": "123321" }' \
    'http://0.0.0.0:8010/api-token-auth/' \
    | sed -e 's/"/ /g' | awk '{print $4}')
echo $token
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MjI0MjM2MDQsImVtYWlsIjoicm9vdEBlbWFpbC5jb20ifQ
↪SdQBLTFvA4qQqKsCyU4Quu2b5VLDjt3QO1b29njfl48
```

4.4 Prepare a Dataset

To use these examples please clone the Network Pipeline Datasets repository locally:

```
git clone https://github.com/jay-johnson/network-pipeline-datasets /opt/antinex/
↪datasets
```

You can also use the AntiNex Datasets repository if you want, they assume you want to build a dataset with the included OWASP fuzzing attack data captured during a ZAP attack simulation in with your own captured CSV files.

4.4.1 Protecting Django with a Deep Neural Network

This guide is a walkthrough for preparing and training a deep neural network for defending Django application servers. The accuracy is currently **70%** without tuning the DNN or adding in actual exploits or sql-injection attacks into the attack datasets.

In the future I am looking to extend the full datasets to include the TCP payload data stream (hex bytes) for sentiment analysis using an embedding Keras layer (https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html). I imagine deserialized payloads will only increase the default accuracy, but it is only an assumption for now.

Setup

1. Run these commands to clone the repositories to the same directories for making debugging easier for all users.

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /
↪opt/antinex/api
```

(continues on next page)

(continued from previous page)

```
git clone https://github.com/jay-johnson/network-pipeline-datasets.git /opt/
↪antinex/datasets
git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/
↪antinex-datasets
```

2. Start the REST API

   If the REST API is not running, please start it in a new terminal so it can process the prepare and training requests.

```
cd /opt/antinex/api
source ~/.venvs/venvdrfpipeline/bin/activate
./install.sh
./start.sh
```

(Optional) Prepare Attack Dataset

If you want to prepare your own attack dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/django/configs/django-
↪attack-prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Check the files were updated:

```
ls -l /opt/antinex/antinex-datasets/v1/webapps/django/inputs/attack/
total 5088
-rw-rw-r-- 1 jay jay    2144 Feb 15 09:22 cleaned_v1_django_attack_metadata.json
-rw-rw-r-- 1 jay jay    2455 Feb 15 09:22 fulldata_v1_django_attack_metadata.json
-rw-rw-r-- 1 jay jay 1131875 Feb 15 09:22 v1_django_cleaned_attack.csv
-rw-rw-r-- 1 jay jay 4064695 Feb 15 09:22 v1_django_full_attack.csv
```

(Optional) Prepare Full Dataset

If you want to prepare your own full dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/django/configs/django-
↪prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Confirm Dataset is Ready

```
/opt/antinex/antinex-datasets/tools/describe-v1-training.py /opt/antinex/antinex-
↪datasets/v1/webapps/django/training-ready/v1_django_cleaned.csv
```

Hopefully your dataset has both attack and non-attack records like:

```
2018-02-15 09:23:23,963 - describe-training-data - INFO - total records=30200␣
↪attack=9000 nonattack=21200 percent_attack=29.80% percent_nonattack=70.20%
```

What you don't want to see is this in the output:

```
2018-02-15 08:47:41,389 - describe-training-data - INFO - total records=21200␣
↪attack=0 nonattack=21200 percent_attack=0.00% percent_nonattack=100.00%
```

That means the prepare step failed to add the attack data into the dataset correctly. Please go back to the `Prepare Dataset` step and review paths to the files are correct.

Train Dataset

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/django/configs/django-train-
↪v1.json
/opt/antinex/api/tests/create-keras-dnn.py
```

From the logs taken during creation of this doc, the model is 70% accurate at predicting attack records.

/opt/antinex/api/tests/create-keras-dnn.py INFO:create-keras-dnn:Logging in user url=http://localhost:8010/api-token-auth/ INFO:create-keras-dnn:logged in user=root token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MTg3MTYwMzk 32Y INFO:create-keras-dnn:building post data INFO:create-keras-dnn:Running ML Job url=http://localhost:8010/ml/ test_data={'csv_file': '/opt/antinex/antinex-datasets/v1/webapps/django/training-ready/v1_django_cleaned.csv', 'meta_file': '/opt/antinex/antinex-datasets/v1/webapps/django/training-ready/cleaned_v1_django_metadata.json', 'title': 'Django - Keras DNN - Dataset v1', 'desc': 'Training Django DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'predict_feature': 'label_value', 'training_data': '{}', 'pre_proc': '{}', 'post_proc': '{}', 'meta_data': '{}', 'version': 1} INFO:create-keras-dnn:SUCCESS - Post Response status=201 reason=Created INFO:create-keras-dnn:{'job': {'id': 14, 'user_id': 1, 'user_name': 'root', 'title': 'Django - Keras DNN - Dataset v1', 'desc': 'Training Django DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'status': 'initial', 'control_state': 'active', 'predict_feature': 'label_value', 'training_data': {}, 'pre_proc': {}, 'post_proc': {}, 'meta_data': {}, 'tracking_id': 'ml_befd247b-7163-4909-87d3-7e43189471a3', 'version': 1, 'created': '2018-02-15 17:28:59', 'updated': '2018-02-15 17:28:59', 'deleted': ''}, 'results': {'id': 10, 'user_id': 1, 'user_name': 'root', 'job_id': 14, 'status': 'finished', 'version': 1, 'acc_data': {**'accuracy': 70.9602648927676**}, 'error_data': None, 'model_json': '{"class_name": "Sequential", "config": [{"class_name": "Dense", "config": {"name": "dense_1", "trainable": true, "batch_input_shape": [null, 68], "dtype": "float32", "units": 8, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name": "dense_2", "trainable": true, "units": 6, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name": "dense_3", "trainable": true, "units": 1, "activation": "sigmoid", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}], "keras_version": "2.1.4", "backend": "tensorflow"}', 'model_weights': {'weights': '[[[0.1209325939,0.125004366,0.04392628,0.0058005759,0.0283837207,0.1084360257,-0.0221137945,0.1256226152],[0.0659536794,0.0419033654,0.0170708448,-0.0357947014,0.0545291603,0.0852750689,-0.0538051911,0.0699182898],[-0.1545251906,0.1111956462,-0.0147213368,0.0731624961,0.0864041299,-0.0017828628,-0.0178274736,0.1216073707],[0.1336804777,0.1023893878,0.0170907527,0.0518929921,0.0966836065,0.159796953
0.0378300808,-0.0057742135,0.0791756362,0.0235338192,0.0824202821],[0.1233218759,0.0594531633,-0.0086939791,-0.0387902856,-0.0360716954,0.0850752816,-0.0114007629,0.0921288431],[-0.1866215467,0.0660640672,-0.0012092546,0.0663910806,0.0470964499,-0.0053454894,-0.0322002359,0.1119380593],[-0.2092835754,0.0618410148,0.0634177029,0.0136457058,0.0184208602,-

```
0.0168401636,-0.0061000162,0.0907672495],[-0.1857144684,0.1216576323,0.0256078299,0.0622631498,0.0746099502,-
0.0211249851,-0.058073774,0.0694563985],[-0.1634339541,0.0696025267,0.0837683603,0.092234768,0.0956235006,-
0.0282980427,-0.040868368,0.113351725],[-0.1704672724,0.0691201314,0.071548,0.069260262,0.1131449491,-
0.0108540468,-0.0425567217,0.0640222654],[-0.2344770879,0.0428572707,0.0793405771,0.0114616835,0.0313309282,0.01253928
0.2056925297,0.0455387048,0.0595687404,0.0481434427,0.0567863956,-0.0198212601,0.0394698866,0.1153029948],[-
0.1900539547,0.0648706928,0.0755348429,0.0531642176,0.0225196443,-0.037876796,-
0.050964918,0.1065156162],[-0.1578101218,0.0851738676,0.080335848,0.1033686772,0.0672588199,-
0.0466286503,0.0274061691,0.1454075873],[-0.1649408489,0.1129061654,0.0413498543,0.0406241789,0.0480666906,-
0.0626311898,-0.0147714363,0.1095488593],[-0.1902400851,0.0546266772,-0.0002129045,0.081249468,0.0261988528,-
0.0456862338,0.033373639,0.1446813047],[-0.2027439624,0.1166701987,0.0509505421,0.0388729684,0.0726077035,-
0.0771305785,0.0307749175,0.0881667733],[-0.183378011,0.0356091969,0.0145586552,0.0244816709,0.0284536369,-
0.0071689375,-0.0315265507,0.1486145407],[-0.1512038261,0.0764901713,0.061774157,0.0451634862,0.062191762,-
0.0525121838,0.0302008335,0.0902092978],[-0.2147508562,0.1103100851,0.017692728,0.0237773284,0.051074788,-
0.0431679115,0.0140604237,0.0951612145],[-0.1971089989,0.0693886876,0.0808551386,0.0206196532,0.0207697973,-
0.039527256,-0.0105302734,0.0831609368],[-0.2245272845,0.1265042126,0.0620332398,0.0791124329,0.0703641251,-
0.0618047714,-0.0481486395,0.1099268496],[-0.1919761449,0.1043844149,0.028273426,0.0244386699,0.0346066169,-
0.0108972676,-0.0562972091,0.1470609605],[-0.1688121408,0.082510218,0.0394482091,0.0875071362,0.1120331734,-
0.0286579132,-0.0526521802,0.0697672367],[-0.1516744047,0.0463477261,0.0772118047,0.0861110315,0.0297314562,-
0.0423166379,-0.006591965,0.0826482922],[-0.2255488038,0.1261228472,0.0011998075,0.0771774799,0.0211141836,-
0.0571117215,0.0122114196,0.1291222423],[-0.2126398236,0.1168462709,0.0215776451,0.0146405725,0.1047314554,-
0.0019828572,-0.000542541,0.0922133848],[0.2239516675,-0.1272274554,-0.0545392334,-0.0690253899,-
0.0240141228,0.0290964283,-0.0048073386,-0.1407266706],[-0.1601528823,0.0873816535,0.0222081486,0.0657906458,0.0444654
0.0557228811,0.0258848574,0.1467054784],[0.1103282571,0.062621966,-0.0022228661,0.0238575842,0.0446507819,0.0041218554
0.009616375,0.0077553303],[-0.190007925,0.0701109022,-0.0119564654,0.0722162873,0.0221073441,-
0.0255252719,-0.0031145217,0.1183288619],[-0.2136628032,0.0534639172,0.0714060888,0.0613537244,0.0842673779,-
0.0301567074,0.0307384171,0.0729675815],[-0.1955242753,0.0495460741,0.0524940416,0.0604530908,0.0595010929,0.016609331
0.0501433276,0.1038881019,0.0417513065,0.0355237499,0.0351028442,-0.0003712648,-
0.0190300811,0.0249827243],[-0.1487564296,0.1013090238,0.0214297064,0.0933733582,0.0616822541,-
0.0710285828,-0.0045771608,0.1217406169],[-0.1543573439,0.0395659693,0.030846579,0.0086379368,0.0861991048,-
0.048596736,-0.0271895695,0.1182741746],[-0.1961124837,0.0571020655,0.0076962849,0.0844913498,0.0193723273,0.006804654
0.0618059374,0.1257505566,0.0142706381,0.1050991416,0.0157455113,-0.0454091579,-
0.0202189703,0.0501390435],[-0.2188960463,0.0440853648,0.07568717,0.0457809009,0.0491918027,-
0.0327213667,-0.0431831405,0.1276204884],[0.1394346952,-0.088656649,-0.0315131843,-0.0401459411,-
0.0437365025,0.0486980379,0.0174318217,-0.0507184826],[-0.1698808372,0.0357401222,0.0317347683,0.0144515438,0.0194635,0
0.0143435514,0.0795696303],[-0.06071667,-0.0370089039,-0.0037398755,0.0016650554,-0.0761372149,-
0.0823373273,-0.0147782043,-0.0841278732],[0.0316180326,-0.0783421397,0.000754284,-
0.0383572131,-0.0244521741,-0.0295800623,-0.0005746271,-0.0404013805],[-0.0592732318,-
0.0370828509,0.0262605324,0.0704324692,0.006007982,-0.117112644,-0.0359724984,0.011283231],[-
0.0184435453,-0.0733112022,-0.0283645988,0.0459322594,-0.0326308981,-0.0342220962,0.0310261045,-
0.1323656887],[-0.0456168354,0.0782427266,0.057690192,0.0134688364,0.0762039647,-0.015021774,-
0.0588690341,0.1178085133],[-0.2152218074,0.0512532629,-0.0107616447,0.0344684459,0.0744112581,0.0065470482,-
0.0018064472,0.1076126173],[-0.1873236597,0.1053858474,-0.011723455,0.058383096,0.1098771915,-
0.0099292547,-0.0258859675,0.0824473128],[-0.1549893022,0.0844767764,0.0727496445,0.0866250396,0.0313318856,-
0.0712586939,0.011944362,0.1336840689],[-0.2329286635,0.1031851396,0.0821548253,0.0666918531,0.0968727544,-
0.0197768845,0.0252549183,0.0636012107],[-0.216961056,0.0476008765,0.041463919,0.0264160633,0.0959576741,-
0.0256117992,-0.0261161607,0.0812392458],[-0.1800720841,0.0473946743,0.0632657334,0.0181837026,0.0440427177,-
0.0560101122,-0.0301355477,0.1441762149],[-0.0096860044,-0.0820260122,-0.0727077052,-0.0164042395,-
0.0778253227,-0.0372423194,-0.2164034247,-0.0453402027],[-0.2012402564,0.0422893129,0.0804941952,0.09306252,0.112802326
0.0451305024,-0.0048267297,0.0649067611],[-0.2305155843,0.0032782811,-0.0235605519,-0.0537962541,-
0.0118097244,-0.0141736846,0.1753456295,0.0044183824],[-0.2079010457,0.069323428,0.0517042466,0.0940786377,-
0.0039390367,-0.0135685029,-0.0057561048,0.0483344011],[-0.179656595,0.0572441481,0.0705757067,0.0079024527,0.024519333
0.0251601636,0.0155755458,0.0752719566],[-0.1713959128,0.1208590493,0.0346451364,0.0616195463,0.0427664891,0.018229700
0.0133397086,0.1184593365],[-0.2043197453,0.0445172973,0.0081551857,0.092943117,0.0198938642,-
0.0723016635,0.0256206822,0.0992775932],[-0.2159956694,0.0412812345,0.0545012057,0.0114885671,0.0877385288,-
```

0.0408159904,0.008009661,0.0881702825],[-0.1577915996,0.0643116087,0.0129699642,0.028255403,0.1068808511,-0.0763266757,0.0346705541,0.0588488467],[-0.2040084004,0.0085537154,-0.0356605425,-0.0487324521,0.0015256398,-0.0052802409,0.1076657921,-0.0199613888],[-0.2131965607,-0.0604146346,-0.0845651999,0.0071733561,-0.0342927612,-0.0888390467,-0.0313468054,-0.0435473919],[-0.0329419039,0.0499187671,0.0159879029,0.0226416737,0.0572009087,0.0169953778,0.0014623989,-0.059061747],[-0.2327010036,0.1027743742,-0.0122413756,0.0495935939,0.0280871764,-0.0538110025,-0.0385680757,0.123318933],[0.0028765725,-0.003429034,0.046994295,0.0192156862,-0.0064633554,-0.0241745599,-0.0367736556,0.0160626136],[-0.0068317289,0.0291511118,0.0879879147,0.0794397742,0.05882008,-0.0122352736,-0.0035967014,-0.0127760237]],[0.1892332435,-0.0836044699,-0.0343620554,-0.0564836971,-0.0631505176,0.0272518657,0.0095205978,-0.1045202538],[[0.1019221023,-0.0305945147,-0.0359275229,0.1524256468,-0.0245757345,0.149545297],[0.0477291718,-0.0069747292,0.0017543581,-0.0294865649,-0.0035721776,-0.0105535956],[0.0457640737,0.0222507585,-0.0189340338,0.0176823959,0.0232398938,0.1038072333],[-0.0229437519,-0.0037464558,-0.0406899974,0.0335939266,-0.0280832294,-0.035844963],[0.0157823972,-0.0069473935,-0.0812087357,0.0187359527,0.0143456571,0.0173410792],[-0.0285115037,0.0277708899,-0.0401631221,0.012221084,-0.0175570287,0.1729588807],[-0.0449816212,0.0595379509,-0.0334494524,0.1174537018,-0.0340393223,-0.0202259049],[-0.001982389,0.008182928,-0.0544667765,-0.0103572421,-0.0074457065,-0.0234164968]],[0.9318162203,-0.0069479314,-0.1054181308,1.2725764513,1.38625741,-0.0264722481],[[-0.2285993248],[-0.0322982036],[-0.0067237676],[-0.3306575119],[-0.9310822487],[-0.0148510356]],[-1.0206410885]]'}, 'acc_image_file': '/media/sf_shared/accuracy_job_14_result_10.png', 'created': '2018-02-15 17:30:21', 'updated': '2018-02-15 17:30:22', 'deleted': ''}}

Get the Deep Neural Network Accuracy, JSON and Weights

This will display all the recent training runs in a list sorted by newest.

```
/opt/antinex/api/tests/get-recent-results.py
```

Here's the training node in the list from the run above (yours will look a little different):

```
{
    "acc_data": {
        "accuracy": 70.9602648927676
    },
    "acc_image_file": "/media/sf_shared/accuracy_job_14_result_10.png",
    "created": "2018-02-15 17:30:21",
    "deleted": "",
    "error_data": null,
    "id": 10,
    "job_id": 14,
    "model_json": "{\"class_name\": \"Sequential\", \"config\": [{\"class_name\": \
↪"Dense\", \"config\": {\"name\": \"dense_1\", \"trainable\": true, \"batch_input_
↪shape\": [null, 68], \"dtype\": \"float32\", \"units\": 8, \"activation\": \"relu\",
↪ \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \"RandomUniform\", \
↪"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}, \"bias_
↪initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_regularizer\":␣
↪null, \"bias_regularizer\": null, \"activity_regularizer\": null, \"kernel_
↪constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\", \
↪"config\": {\"name\": \"dense_2\", \"trainable\": true, \"units\": 6, \"activation\
↪": \"relu\", \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \
↪"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}
↪, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_
↪regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\": null, \
↪"kernel_constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\",
↪ \"config\": {\"name\": \"dense_3\", \"trainable\": true, \"units\": 1, \
↪"activation\": \"sigmoid\", \"use_bias\": true, \"kernel_initializer\": {\"class_
↪name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\
↪": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \
↪"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\":␣
↪null, \"kernel_constraint\": null, \"bias_constraint\": null}}], \"keras_version\":␣
↪\"2.1.4\", \"backend\": \"tensorflow\"}",
```

(continues on next page)

(continued from previous page)

```
    "model_weights": {
        "weights": "[[[0.1209325939,0.125004366,0.04392628,0.0058005759,0.0283837207,
↪0.1084360257,-0.0221137945,0.1256226152],[0.0659536794,0.0419033654,0.0170708448,-0.
↪0357947014,0.0545291603,0.0852750689,-0.0538051911,0.0699182898],[-0.1545251906,0.
↪1111956462,-0.0147213368,0.0731624961,0.0864041299,-0.0017828628,-0.0178274736,0.
↪1216073707],[0.1336804777,0.1023893878,0.0170907527,0.0518929921,0.0966836065,0.
↪1597969532,0.0217716675,0.085526742],[0.1376502961,0.0593057014,0.0694345012,-0.
↪0378300808,-0.0057742135,0.0791756362,0.0235338192,0.0824202821],[0.1233218759,0.
↪0594531633,-0.0086939791,-0.0387902856,-0.0360716954,0.0850752816,-0.014007629,0.
↪0921288431],[-0.1866215467,0.0660640672,-0.0012092546,0.0663910806,0.0470964499,-0.
↪0053454894,-0.0322002359,0.1119380593],[-0.2092835754,0.0618410148,0.0634177029,0.
↪0136457058,0.0184208602,-0.0168401636,-0.0061000162,0.0907672495],[-0.1857144684,0.
↪1216576323,0.0256078299,0.0622631498,0.0746099502,-0.0211249851,-0.058073774,0.
↪0694563985],[-0.1634339541,0.0696025267,0.0837683603,0.092234768,0.0956235006,-0.
↪0282980427,-0.040868368,0.113351725],[-0.1704672724,0.0691201314,0.071548,0.
↪069260262,0.1131449491,-0.0108540468,-0.0425567217,0.0640222654],[-0.2344770879,0.
↪0428572707,0.0793405771,0.0114616835,0.0313309282,0.0125392843,0.0246058684,0.
↪0830053911],[-0.2056925297,0.0455387048,0.0595687404,0.0481434427,0.0567863956,-0.
↪0198212601,0.0394698866,0.1153029948],[-0.1900539547,0.0648706928,0.0755348429,0.
↪0531642176,0.0225196443,-0.037876796,-0.050964918,0.1065156162],[-0.1578101218,0.
↪0851738676,0.080335848,0.1033686772,0.0672588199,-0.0466286503,0.0274061691,0.
↪1454075873],[-0.1649408489,0.1129061654,0.0413498543,0.0406241789,0.0480666906,-0.
↪0626311898,-0.0147714363,0.1095488593],[-0.1902400851,0.0546266772,-0.0002129045,0.
↪081249468,0.0261988528,-0.0456862338,0.033373639,0.1446813047],[-0.2027439624,0.
↪1166701987,0.0509505421,0.0388729684,0.0726077035,-0.0771305785,0.0307749175,0.
↪0881667733],[-0.183378011,0.0356091969,0.0145586552,0.0244816709,0.0284536369,-0.
↪0071689375,-0.0315265507,0.1486145407],[-0.1512038261,0.0764901713,0.061774157,0.
↪0451634862,0.062191762,-0.0525121838,0.0302008335,0.0902092978],[-0.2147508562,0.
↪1103100851,0.017692728,0.0237773284,0.051074788,-0.0431679115,0.0140604237,0.
↪0951612145],[-0.1971089989,0.0693886876,0.0808551386,0.0206196532,0.0207697973,-0.
↪039527256,-0.0105302734,0.0831609368],[-0.2245272845,0.1265042126,0.0620332398,0.
↪0791124329,0.0703641251,-0.0618047714,-0.0481486395,0.1099268496],[-0.1919761449,0.
↪1043844149,0.028273426,0.0244386699,0.0346066169,-0.0108972676,-0.0562972091,0.
↪1470609605],[-0.1688121408,0.082510218,0.0394482091,0.0875071362,0.1120331734,-0.
↪0286579132,-0.0526521802,0.0697672367],[-0.1516744047,0.0463477261,0.0772118047,0.
↪0861110315,0.0297314562,-0.0423166379,-0.006591965,0.0826482922],[-0.2255488038,0.
↪1261228472,0.0011998075,0.0771774799,0.0211141836,-0.0571117215,0.0122114196,0.
↪1291222423],[-0.2126398236,0.1168462709,0.0215776451,0.0146405725,0.1047314554,-0.
↪0019828572,-0.000542541,0.0922133848],[0.2239516675,-0.1272274554,-0.0545392334,-0.
↪0690253899,-0.0240141228,0.0290964283,-0.0048073386,-0.1407266706],[-0.1601528823,0.
↪0873816535,0.0222081486,0.0657906458,0.0444654338,-0.0557228811,0.0258848574,0.
↪1467054784],[0.1103282571,0.062621966,-0.0022228661,0.0238575842,0.0446507819,0.
↪0041218554,-0.009616375,0.0077553303],[-0.190007925,0.0701109022,-0.0119564654,0.
↪0722162873,0.0221073441,-0.0255252719,-0.0031145217,0.1183288619],[-0.2136628032,0.
↪0534639172,0.0714060888,0.0613537244,0.0842673779,-0.0301567074,0.0307384171,0.
↪0729675815],[-0.1955242753,0.0495460741,0.0524940416,0.0604530908,0.0595010929,0.
↪0166093316,0.0391650833,0.0827598944],[-0.0501433276,0.1038881019,0.0417513065,0.
↪0355237499,0.0351028442,-0.0003712648,-0.0190300811,0.0249827243],[-0.1487564296,0.
↪1013090238,0.0214297064,0.0933733582,0.0616822541,-0.0710285828,-0.0045771608,0.
↪1217406169],[-0.1543573439,0.0395659693,0.030846579,0.0086379368,0.0861991048,-0.
↪048596736,-0.0271895695,0.1182741746],[-0.1961124837,0.0571020655,0.0076962849,0.
↪0844913498,0.0193723273,0.0068046544,0.0302888621,0.143293947],[-0.0618059374,0.
↪1257505566,0.0142706381,0.1050991416,0.0157455113,-0.0454091579,-0.0202189703,0.
↪0501390435],[-0.2188960463,0.0440853648,0.07568717,0.0457809009,0.0491918027,-0.
↪0327213667,-0.0431831405,0.1276204884],[0.1394346952,-0.088656649,-0.0315131843,-0.
↪0401459411,-0.0437365025,0.0486980379,0.0174318217,-0.0507184826],[-0.1698808372,0.
↪0357401222,0.0317347683,0.0144515438,0.0194635,0.0180209279,-0.0143435514,0.
↪0795696303],[-0.06071667,-0.0370089039,-0.0037398755,0.0016650554,-0.07[illegible],0.
↪0823373273,-0.0147782043,-0.0841278732],[0.0316180326,-0.0783421397,0.000754284,-0.
↪0383572131,-0.0244521741,-0.0295800623,-0.0005746271,-0.0404013805],[-0.0592732318,-
↪0.0370828509,0.0262605324,0.0704324692,0.006007982,-0.117112644,-0.0359724984,0.
↪011283231],[-0.0184435453,-0.0733112022,-0.0283645988,0.0459322594,-0.0326308981,-0.
↪0342220962,0.0310261045,-0.1323656887],[-0.0456168354,0.0782427266,0.057690192,0.
↪0134688364,0.0762039647,-0.015021774,-0.0588690341,0.1178085133],[-0.2152218074,0.
```

(continues on next page)

(continued from previous page)

```
    },
    "status": "finished",
    "updated": "2018-02-15 17:30:22",
    "user_id": 1,
    "user_name": "root",
    "version": 1
}
```

4.4.2 Protecting Flask RESTplus with a Deep Neural Network

This guide is a walkthrough for preparing and training a deep neural network for defending Flask RESTplus application servers. The accuracy is currently **89%** without tuning the DNN or adding in actual exploits or sql-injection attacks into the attack datasets. Please note the `non-attack` training data is recorded from a multi-user simulation against a Django application server. Sorry I have not had enough free time to create a true Flask non-attack dataset (PRs welcome though!).

In the future I am looking to extend the full datasets to include the TCP payload data stream (hex bytes) for sentiment analysis using an embedding Keras layer (https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html). I imagine deserialized payloads will only increase the default accuracy, but it is only an assumption for now.

Setup

1. Run these commands to clone the repositories to the same directories for making debugging easier for all users.

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /
↪opt/antinex/api
git clone https://github.com/jay-johnson/network-pipeline-datasets.git /opt/
↪antinex/datasets
git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/
↪antinex-datasets
```

2. Start the REST API

If the REST API is not running, please start it in a new terminal so it can process the prepare and training requests.

```
cd /opt/antinex/api
source ~/.venvs/venvdrfpipeline/bin/activate
./install.sh
./start.sh
```

(Optional) Prepare Attack Dataset

If you want to prepare your own attack dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/flask-restplus/configs/
↪flask-attack-prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Check the files were updated:

```
ls -l /opt/antinex/antinex-datasets/v1/webapps/flask-restplus/inputs/attack/
total 1052
-rw-rw-r-- 1 jay jay   2088 Feb 15 10:54 cleaned_v1_flask_attack_metadata.json
-rw-rw-r-- 1 jay jay   2359 Feb 15 10:54 fulldata_v1_flask_attack_metadata.json
-rw-rw-r-- 1 jay jay 322110 Feb 15 10:54 v1_flask_cleaned_attack.csv
-rw-rw-r-- 1 jay jay 745217 Feb 15 10:54 v1_flask_full_attack.csv
```

(Optional) Prepare Full Dataset

If you want to prepare your own full dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/flask-restplus/configs/
↪flask-prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Confirm Dataset is Ready

```
/opt/antinex/antinex-datasets/tools/describe-v1-training.py /opt/antinex/antinex-
↪datasets/v1/webapps/flask-restplus/training-ready/v1_flask_cleaned.csv
```

Hopefully your dataset has both attack and non-attack records like:

```
2018-02-15 10:57:05,417 - describe-training-data - INFO - total records=23800␣
↪attack=2600 nonattack=21200 percent_attack=10.92% percent_nonattack=89.08%
```

What you don't want to see is this in the output:

```
2018-02-15 08:47:41,389 - describe-training-data - INFO - total records=21200␣
↪attack=0 nonattack=21200 percent_attack=0.00% percent_nonattack=100.00%
```

That means the prepare step failed to add the attack data into the dataset correctly. Please go back to the `Prepare Dataset` step and review paths to the files are correct.

Train Dataset

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/flask-restplus/configs/
↪flask-train-v1.json
/opt/antinex/api/tests/create-keras-dnn.py
```

From the logs taken during creation of this doc, the model is 89% accurate at predicting attack records.

/opt/antinex/api/tests/create-keras-dnn.py INFO:create-keras-dnn:Logging in user url=http://localhost:8010/api-token-auth/ INFO:create-keras-dnn:logged in user=root token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MTg3MjEzNDUsIn INFO:create-keras-dnn:building post data INFO:create-keras-dnn:Running ML Job url=http://localhost:8010/ml/ test_data={'csv_file': '/opt/antinex/antinex-datasets/v1/webapps/flask-restplus/training-ready/v1_flask_cleaned.csv', 'meta_file': '/opt/antinex/antinex-datasets/v1/webapps/flask-restplus/training-ready/cleaned_v1_flask_metadata.json', 'title': 'Flask RESTplus - Keras DNN - Dataset v1', 'desc': 'Training Flask RESTplus DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'predict_feature': 'label_value', 'training_data': '{}', 'pre_proc': '{}',

'post_proc': '{}', 'meta_data': '{}', 'version': 1} INFO:create-keras-dnn:SUCCESS - Post Response status=201 reason=Created INFO:create-keras-dnn:{'job': {'id': 15, 'user_id': 1, 'user_name': 'root', 'title': 'Flask RESTplus - Keras DNN - Dataset v1', 'desc': 'Training Flask RESTplus DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'status': 'initial', 'control_state': 'active', 'predict_feature': 'label_value', 'training_data': {}, 'pre_proc': {}, 'post_proc': {}, 'meta_data': {}, 'tracking_id': 'ml_185aadad-7df6-4f87-92f8-8d7cfe9ccaa1', 'version': 1, 'created': '2018-02-15 18:57:25', 'updated': '2018-02-15 18:57:25', 'deleted': ''}, 'results': {'id': 11, 'user_id': 1, 'user_name': 'root', 'job_id': 15, 'status': 'finished', 'version': 1, 'acc_data': {'**accuracy': 89.49579831932773**}, 'error_data': None, 'model_json': '{"class_name": "Sequential", "config": [{"class_name": "Dense", "config": {"name": "dense_1", "trainable": true, "batch_input_shape": [null, 68], "dtype": "float32", "units": 8, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name": "dense_2", "trainable": true, "units": 6, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name": "dense_3", "trainable": true, "units": 1, "activation": "sigmoid", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}], "keras_version": "2.1.4", "backend": "tensorflow"}', 'model_weights': {'weights': '[[[0.0426323749,-0.0255882181,-
0.0090597291,0.1283773184,0.0292089302,0.0025187351,0.0829728991,0.1176706031],[0.037888933,0.0087408721,0.0459016114,
0.016116729,0.0464626513,0.0540330783,0.022757668],[-0.0214296542,0.0262879357,0.0038285167,-
0.0734874904,-0.0107907392,0.0001898558,-0.1048046276,-0.0412591994],[0.007992073,-
0.0265324973,0.0369161405,0.0394662991,0.0031596741,-0.0152786113,0.0967443436,0.0810386315],[0.0589824058,0.006413919
0.0196403004,0.1306117624,0.0538136661,-0.0151381232,0.1598619968,0.1096331924],[-
0.0228838958,0.0238968544,0.0322253667,-0.0315970778,0.0328627527,0.0515150316,-0.1444948912,-
0.0172979422],[-0.0669921488,-0.0633563995,0.026413843,-0.0758443102,0.0212015193,0.0008845639,-
0.1340515614,-0.0885572657],[-0.0379409157,0.0235466771,-0.0239608623,-0.0233518947,-
0.0180600733,-0.000621935,-0.0909899473,-0.0650934204],[-0.1045127213,0.0289334524,-
0.0202295408,-0.1071161181,0.0011363167,0.0574624874,-0.1407585889,-0.1154722199],[-
0.0820850208,-0.0178494379,-0.0217043143,-0.0971601605,-0.0540295169,0.0126081835,-
0.1323711425,-0.0351396762],[-0.0396530814,-0.0489722192,-0.0246039722,-0.0223898534,-
0.0241458677,0.034281414,-0.1406630576,-0.1091260538],[-0.0556271039,-0.0563059077,-
0.0233718585,-0.0885359943,-0.0523045808,0.0029973972,-0.047497876,-0.0678029805],[-0.0830031335,-
0.0494688451,0.0220204517,-0.068020612,0.0165492371,0.0551169775,-0.0784559101,-0.0418072231],[-
0.0427264832,-0.0430617929,0.0010199838,-0.0166956335,0.0032942081,0.001454784,-0.0962905437,-
0.0894291624],[-0.0758964419,-0.0236174613,0.0260267798,-0.1116409451,-0.039193593,-0.0216977205,-
0.1275075376,-0.0769466385],[-0.0189716518,-0.0011924787,0.0378293768,-0.0616473816,-
0.0058780708,-0.0095390407,-0.0897467956,-0.0901271477],[-0.0807787329,-0.0143492874,-
0.0354749262,-0.0797566995,0.0338315442,-0.024086047,-0.0861588418,-0.0659692138],[-0.0624345131,-
0.0491259694,0.0337389037,-0.0533887483,0.0058928118,-0.035079021,-0.134863764,-0.0412503555],[-
0.0379889719,-0.0485685989,0.0187181961,-0.0228507519,-0.051875487,0.0183100309,-0.1258815378,-
0.0734021738],[-0.0277079009,0.0286247917,0.0038159075,-0.0966584384,-0.005367511,0.0266397614,-
0.1313499063,-0.0257532299],[-0.0770806,-0.0542856306,-0.000666048,-0.0634321198,0.026355302,-
0.0066100159,-0.0456909463,-0.0334356874],[-0.0904787034,-0.0271485578,0.0489063449,-
0.0324975327,-0.0418805331,0.0154382363,-0.0966121927,-0.0713369027],[-0.0432081558,-
0.0086686136,-0.0281147528,-0.0924807042,0.0139136631,-0.0166061427,-0.0506068021,-
0.0763159022],[-0.0625338033,0.0062450422,0.0534292571,-0.0707026199,-0.024397444,-0.0328290649,-
0.1001479179,-0.0974239931],[-0.0936535373,-0.0477943867,-0.0313425213,-0.1001306772,-
0.0306321867,-0.0217516664,-0.1106291041,-0.023179628],[-0.1038499326,-0.028331412,-
0.0345370658,-0.0847722739,0.0043513202,-0.0235293116,-0.1417917758,-0.1084775403],[-
0.0255202819,0.0110019408,0.0376705453,-0.0307282936,-0.0009297428,-0.0346140675,-

0.0557651855,-0.075329639],[0.0327686854,-0.0048621781,-0.0449326374,0.1063804403,0.0354809873,-
0.031704843,0.1717805862,0.0739540011],[-0.0120043308,0.0316974148,0.0294647831,-0.0407527275,-
0.0047504301,-0.0305531397,-0.0991907716,-0.0953953043],[-0.1024203598,0.099888809,-0.0079205092,-
0.0490499474,-0.0271461513,0.0822322369,0.0575119071,0.0676374361],[-0.0975352377,-0.0279858597,-
0.0375377163,-0.0733665898,0.0024956453,0.0000131985,-0.0623091795,-0.0348203406],[-
0.1044042036,-0.0493010655,0.0560958013,-0.0540509932,-0.0211698078,-0.0265545212,-0.0715102479,-
0.0359795876],[-0.0425768159,0.0305278897,-0.0213658791,-0.0220552664,-0.0028750291,0.0152840279,-
0.1093827412,-0.0265729669],[-0.0947411284,0.0571837798,0.0370233022,-0.0815027654,-
0.0462668501,0.0736624524,-0.0058687618,-0.0746894926],[-0.0609302185,0.0090621058,-
0.0419382341,-0.1110604107,-0.0119013209,-0.017867554,-0.0557432212,-0.087935932],[-
0.0381675027,0.0101275556,0.0067694304,-0.0998164713,-0.0359654427,-0.0082442584,-0.0981924981,-
0.1116829589],[-0.0741901025,0.0291563496,-0.0264984984,-0.0616867431,-0.0063055283,0.0279856529,-
0.0907634646,-0.1027278528],[-0.0758224577,0.0015842745,0.0338854305,-0.0514345653,-0.010851006,-
0.0060120882,-0.0644992739,-0.0726546496],[-0.0508742668,-0.0639033169,-0.0423267372,-
0.0164754614,0.0243874229,-0.0255884398,-0.1314629316,-0.0759309903],[0.0438570306,-
0.0348124057,-0.0189036988,0.016748948,0.0424729399,-0.0581890643,0.0896789953,0.0363361128],[-
0.0386991538,-0.0180182755,-0.0424260609,-0.1039735526,0.0112775657,0.0560146682,-
0.1352385879,-0.0222511459],[-0.035192512,-0.0047485712,0.015257326,-0.0013657424,-0.0506449156,-
0.0291663408,0.0103004025,0.0294213798],[0.0089905122,0.0350620709,0.0046768119,-0.0415131934,-
0.0150964623,0.0221417733,-0.0056831283,0.0544817522],[-0.0420532711,0.0079685589,0.0067148004,-
0.082384862,0.0022194732,-0.0165153295,-0.1248686388,-0.019711487],[-0.096077241,-0.0254066326,-
0.0540779792,-0.0449774973,-0.0033162525,0.0440527797,-0.0071636667,-0.0076389913],[0.0116391173,-
0.0320617296,-0.0108245742,0.0027867109,0.0180245712,0.0012698642,-0.099055782,-
0.0219898615],[0.00900253,0.0287133083,0.0537100956,-0.0744302124,0.0305081364,0.0016400049,-
0.0700737685,-0.0486903861],[0.0033208129,0.0157403499,-0.0253775604,-0.0587556846,0.0218016598,0.028432576,-
0.1294285953,-0.0836849883],[0.0367537104,-0.0175779108,-0.0051506036,-0.057613682,0.0313959233,-
0.0433652923,-0.0949594006,-0.0432897843],[-0.0339793079,0.0020113038,0.0124180513,-
0.0530905798,-0.0427044183,-0.0553700179,-0.1070849448,-0.0691610053],[0.0001814616,-
0.0168834105,0.0339019485,-0.0775102973,-0.0191271659,0.0216677003,-0.0656533465,-
0.1144009531],[-0.1059497893,-0.0049181273,0.0498044118,-0.0346838795,0.0164915081,0.0254156869,-
0.1205460355,-0.0772420093],[-0.0380106792,-0.0061189532,-0.0359303355,-0.0459102169,-
0.0334480405,-0.0367637649,0.0358054154,0.0131852068],[-0.0965600684,0.0022080662,-
0.0093741212,-0.033005625,-0.0194996297,0.0079977531,-0.1066729948,-0.0651928782],[-
0.0560397878,0.0328344405,-0.0274874251,0.0420479812,0.0315765962,0.0070940158,-
0.0787004307,0.0073882868],[-0.0930133164,-0.0196791496,-0.0011561333,-0.0399722718,-
0.0479589812,0.0000561367,-0.0738650635,-0.0443399698],[-0.0840743333,-0.0551205203,0.0241359416,-
0.0662114322,0.0348841324,-0.003500547,-0.125039205,-0.0998590812],[-0.0498304553,0.03142488,-
0.0002348951,-0.0582877211,0.0119131543,0.0372843482,-0.1186875254,-0.0261092521],[-
0.0891042799,-0.0493348017,0.0338282734,-0.0400045402,0.0408838838,-0.0235499088,-0.0468597487,-
0.0568073764],[-0.0808104947,-0.0333214775,-0.0042998446,-0.065806143,0.0337486975,0.0285946317,-
0.132270664,-0.0551608354],[-0.0990937799,0.0321997777,0.0339404307,-0.0641544685,0.029329313,-
0.0001243317,-0.111442402,-0.026380565],[-0.0159308631,-0.008656092,-0.0455170162,0.0365302898,-
0.0284806192,0.0253368132,-0.1148158312,-0.0022615485],[0.0044460897,0.053295508,0.0307782292,-
0.0550564155,0.0170134939,-0.0393051617,-0.1321098059,0.0248682518],[0.0363558196,-
0.0131116873,0.0325888433,0.0374935232,-0.028289672,-0.0231639612,-0.0569625199,0.0362283513],[-
0.0320892818,-0.0178287178,-0.0314985737,-0.023245478,0.0207714792,0.0148529997,-0.1123319864,-
0.0739320144],[-0.0294432193,0.0545074418,0.0186652243,-0.0806119889,0.0009009295,0.025605455,0.0006349441,-
0.0063184882],[0.0505009703,0.0361768678,0.0135121401,-0.0116158333,-0.0491209626,-0.0255839732,-
0.0957046375,-0.0473603681]],[0.0588490553,0.0163822807,-0.0064409007,0.0641452521,0.0059181629,-
0.0099197142,0.0951542407,0.0660104603],[[-0.0050316565,0.0507251173,-0.0241616126,0.0318146385,-
0.0744105875,0.0357613154],[0.0290509574,0.0377634503,-0.0239083767,0.0000599554,-
0.0120540028,0.042443905],[0.0221073218,0.0118085258,0.0258703269,0.0051663192,-0.0452455804,-
0.0262466893],[-0.0048780073,0.0732062235,-0.0393346548,-0.0642782897,0.0263430309,-
0.0082829352],[-0.0472923033,0.0224473402,-0.0167164803,0.0103773978,0.0015468168,-

0.0262690522],[-0.0477254875,0.0187563188,-0.0213371273,-0.0300161056,0.023460472,0.0163392462],[-0.0863996446,0.1024220511,0.035894502,-0.0403372794,-0.0435755812,0.1845877171],[-0.0398054905,0.0403691866,-0.0404988527,-0.0220432878,0.0178212691,0.06267111]],[-0.0122304028,0.0925562233,0.0,-0.0065102046,-0.0187473632,0.0327210948],[[0.022001354],[-0.0505836755],[-0.015696872],[0.0343161002],[0.0277610142],[-0.0699139088]],[-0.072642006]]'}, 'acc_image_file': '/media/sf_shared/accuracy_job_15_result_11.png', 'created': '2018-02-15 18:58:29', 'updated': '2018-02-15 18:58:30', 'deleted': ''}}

Get the Deep Neural Network Accuracy, JSON and Weights

This will display all the recent training runs in a list sorted by newest.

```
/opt/antinex/api/tests/get-recent-results.py
```

Here's the training node in the list from the run above (yours will look a little different):

```
{
    "acc_data": {
        "accuracy": 89.49579831932773
    },
    "acc_image_file": "/media/sf_shared/accuracy_job_15_result_11.png",
    "created": "2018-02-15 18:58:29",
    "deleted": "",
    "error_data": null,
    "id": 11,
    "job_id": 15,
    "model_json": "{\"class_name\": \"Sequential\", \"config\": [{\"class_name\": \
↪"Dense\", \"config\": {\"name\": \"dense_1\", \"trainable\": true, \"batch_input_
↪shape\": [null, 68], \"dtype\": \"float32\", \"units\": 8, \"activation\": \"relu\",
↪ \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \"RandomUniform\", \
↪"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}, \"bias_
↪initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_regularizer\":␣
↪null, \"bias_regularizer\": null, \"activity_regularizer\": null, \"kernel_
↪constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\", \
↪"config\": {\"name\": \"dense_2\", \"trainable\": true, \"units\": 6, \"activation\
↪": \"relu\", \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \
↪"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}
↪, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_
↪regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\": null, \
↪"kernel_constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\",
↪ \"config\": {\"name\": \"dense_3\", \"trainable\": true, \"units\": 1, \
↪"activation\": \"sigmoid\", \"use_bias\": true, \"kernel_initializer\": {\"class_
↪name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\
↪": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \
↪"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\":␣
↪null, \"kernel_constraint\": null, \"bias_constraint\": null}}], \"keras_version\":␣
↪\"2.1.4\", \"backend\": \"tensorflow\"}",
    "model_weights": {
        "weights": "[[[0.0426323749,-0.0255882181,-0.0090597291,0.1283773184,0.
↪0292089302,0.0025187351,0.0829728991,0.1176706031],[0.037888933,0.0087408721,0.
↪0459016114,0.1122659445,-0.016116729,0.0464626513,0.0540330783,0.022757668],[-0.
↪0214296542,0.0262879357,0.0038285167,-0.0734874904,-0.0107907392,0.0001898558,-0.
↪1048046276,-0.0412591994],[0.007992073,-0.0265324973,0.0369161405,0.0394662991,0.
↪0031596741,-0.0152786113,0.0967443436,0.0810386315],[0.0589824058,0.0064139194,0.
↪0142697673,0.1090230495,0.0537979342,0.0046058199,0.1009583548,0.0961056277],[0.
↪0956505686,0.0055070231,-0.0196403004,0.1306117624,0.0538136661,-0.0151381232,0.
↪1598619968,0.1096331924],[-0.0228838958,0.0238968544,0.0322253667,-0.0315970778,0.
↪0328627527,0.0515150316,-0.1444948912,-0.0172979422],[-0.0669921488,-0.0653563995,0.
↪026413843,-0.0758443102,0.0212015193,0.0008845639,-0.1340515614,-0.0885572657],[-0.
↪0379409157,0.0235466771,-0.0239608623,-0.0233518947,-0.0180600733,-0.000621935,-0.
↪0909899473,-0.0650934204],[-0.1045127213,0.0289334524,-0.0202295408,-0.1071161181,0.
↪0011363167,0.0574624874,-0.1407585889,-0.1154722199],[-0.0820850208,-0.0178494379,-
↪0.0217043143,-0.0971601605,-0.0540295169,0.0126081835,-0.1323711425,-0.0351396762],
↪[-0.0396530814,-0.0489722192,-0.0246039722,-0.0223898534,-0.0241458677,0.034281414,-
```

(continues on next page)

(continued from previous page)

```
    },
    "status": "finished",
    "updated": "2018-02-15 18:58:30",
    "user_id": 1,
    "user_name": "root",
    "version": 1
}
```

4.4.3 Protecting React and Redux with a Deep Neural Network

This guide is a walkthrough for preparing and training a deep neural network for defending React and Redux application servers. The accuracy is currently **87%** without tuning the DNN or adding in actual exploits or sql-injection attacks into the attack datasets. Please note the `non-attack` training data is recorded from a multi-user simulation against a Django application server. Sorry I have not had enough free time to create a true React and Redux non-attack dataset (PRs welcome though!).

In the future I am looking to extend the full datasets to include the TCP payload data stream (hex bytes) for sentiment analysis using an embedding Keras layer (https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html). I imagine deserialized payloads will only increase the default accuracy, but it is only an assumption for now.

Setup

1. Run these commands to clone the repositories to the same directories for making debugging easier for all users.

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /
↪opt/antinex/api
git clone https://github.com/jay-johnson/network-pipeline-datasets.git /opt/
↪antinex/datasets
git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/
↪antinex-datasets
```

2. Start the REST API

   If the REST API is not running, please start it in a new terminal so it can process the prepare and training requests.

```
cd /opt/antinex/api
source ~/.venvs/venvdrfpipeline/bin/activate
./install.sh
./start.sh
```

(Optional) Prepare Attack Dataset

If you want to prepare your own attack dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/react-redux/configs/react-
↪attack-prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Check the files were updated:

```
ls -l /opt/antinex/antinex-datasets/v1/webapps/react-redux/inputs/attack/
total 4180
-rw-rw-r-- 1 jay jay    2085 Feb 15 11:08 cleaned_v1_react_attack_metadata.json
-rw-rw-r-- 1 jay jay    2416 Feb 15 11:08 fulldata_v1_react_attack_metadata.json
-rw-rw-r-- 1 jay jay  383193 Feb 15 11:08 v1_react_cleaned_attack.csv
-rw-rw-r-- 1 jay jay 3883685 Feb 15 11:08 v1_react_full_attack.csv
```

(Optional) Prepare Full Dataset

If you want to prepare your own full dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/react-redux/configs/react-
↪prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Confirm Dataset is Ready

```
/opt/antinex/antinex-datasets/tools/describe-v1-training.py /opt/antinex/antinex-
↪datasets/v1/webapps/react-redux/training-ready/v1_react_cleaned.csv
```

Hopefully your dataset has both attack and non-attack records like:

```
2018-02-15 11:09:39,541 - describe-training-data - INFO - total records=24300␣
↪attack=3100 nonattack=21200 percent_attack=12.76% percent_nonattack=87.24%
```

What you don't want to see is this in the output:

```
2018-02-15 08:47:41,389 - describe-training-data - INFO - total records=21200␣
↪attack=0 nonattack=21200 percent_attack=0.00% percent_nonattack=100.00%
```

That means the prepare step failed to add the attack data into the dataset correctly. Please go back to the `Prepare Dataset` step and review paths to the files are correct.

Train Dataset

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/react-redux/configs/react-
↪train-v1.json
/opt/antinex/api/tests/create-keras-dnn.py
```

From the logs taken during creation of this doc, the model is 87% accurate at predicting attack records.

/opt/antinex/api/tests/create-keras-dnn.py INFO:create-keras-dnn:Logging in user url=http://localhost:8010/api-token-auth/ INFO:create-keras-dnn:logged in user=root token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MTg3MjIxMDAs hlDpCDxrltx50FFAjMPunberb5GYI748 INFO:create-keras-dnn:building post data INFO:create-keras-dnn:Running ML Job url=http://localhost:8010/ml/ test_data={'csv_file': '/opt/antinex/antinex-datasets/v1/webapps/react-redux/training-ready/v1_react_cleaned.csv', 'meta_file': '/opt/antinex/antinex-datasets/v1/webapps/react-redux/training-ready/cleaned_v1_react_metadata.json', 'title': 'React and Redux - Keras DNN - Dataset v1', 'desc': 'Training React and Redux DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'predict_feature': 'label_value', 'training_data': '{}',

‘pre_proc’: ‘{}’, ‘post_proc’: ‘{}’, ‘meta_data’: ‘{}’, ‘version’: 1} INFO:create-keras-dnn:SUCCESS - Post Response status=201 reason=Created INFO:create-keras-dnn:{‘job’: {‘id’: 16, ‘user_id’: 1, ‘user_name’: ‘root’, ‘title’: ‘React and Redux - Keras DNN - Dataset v1’, ‘desc’: ‘Training React and Redux DNN using Attack and Non-attack data captured using the network-pipeline’, ‘ds_name’: ‘cleaned’, ‘algo_name’: ‘dnn’, ‘ml_type’: ‘keras’, ‘status’: ‘initial’, ‘control_state’: ‘active’, ‘predict_feature’: ‘label_value’, ‘training_data’: {}, ‘pre_proc’: {}, ‘post_proc’: {}, ‘meta_data’: {}, ‘tracking_id’: ‘ml_99743fd9-ff52-4bf7-b42e-3ec25434507d’, ‘version’: 1, ‘created’: ‘2018-02-15 19:10:00’, ‘updated’: ‘2018-02-15 19:10:00’, ‘deleted’: ‘’}, ‘results’: {‘id’: 12, ‘user_id’: 1, ‘user_name’: ‘root’, ‘job_id’: 16, ‘status’: ‘finished’, ‘version’: 1, ‘acc_data’: {**‘accuracy’: 87.4074073926902**}, ‘error_data’: None, ‘model_json’: ‘{“class_name”: “Sequential”, “config”: [{“class_name”: “Dense”, “config”: {“name”: “dense_4”, “trainable”: true, “batch_input_shape”: [null, 68], “dtype”: “float32”, “units”: 8, “activation”: “relu”, “use_bias”: true, “kernel_initializer”: {“class_name”: “RandomUniform”, “config”: {“minval”: -0.05, “maxval”: 0.05, “seed”: null}}, “bias_initializer”: {“class_name”: “Zeros”, “config”: {}}, “kernel_regularizer”: null, “bias_regularizer”: null, “activity_regularizer”: null, “kernel_constraint”: null, “bias_constraint”: null}}, {“class_name”: “Dense”, “config”: {“name”: “dense_5”, “trainable”: true, “units”: 6, “activation”: “relu”, “use_bias”: true, “kernel_initializer”: {“class_name”: “RandomUniform”, “config”: {“minval”: -0.05, “maxval”: 0.05, “seed”: null}}, “bias_initializer”: {“class_name”: “Zeros”, “config”: {}}, “kernel_regularizer”: null, “bias_regularizer”: null, “activity_regularizer”: null, “kernel_constraint”: null, “bias_constraint”: null}}, {“class_name”: “Dense”, “config”: {“name”: “dense_6”, “trainable”: true, “units”: 1, “activation”: “sigmoid”, “use_bias”: true, “kernel_initializer”: {“class_name”: “RandomUniform”, “config”: {“minval”: -0.05, “maxval”: 0.05, “seed”: null}}, “bias_initializer”: {“class_name”: “Zeros”, “config”: {}}, “kernel_regularizer”: null, “bias_regularizer”: null, “activity_regularizer”: null, “kernel_constraint”: null, “bias_constraint”: null}}], “keras_version”: “2.1.4”, “backend”: “tensorflow”}’, ‘model_weights’: {‘weights’: ‘[[[0.052631028,-0.0195708107,0.0235880036,-0.0160469897,0.0599223375,-0.019402137,-
0.0315921232,-0.0252120867],[-0.0266926326,-0.0446610935,0.0770860612,-0.0265123285,-0.0030328943,-
0.0082395915,-0.0333385319,-0.0549867488],[-0.0178412981,-0.0179316401,-0.0041324655,0.0311650522,-
0.0275568571,-0.0177768506,-0.0017130028,0.0179622211],[-0.0210418832,0.0191573389,-
0.0079981312,-0.0045636012,-0.0231961794,-0.0242097769,-0.0480531305,0.0070477622],[-
0.0076428168,-0.0474744439,0.0809031352,0.0383856446,0.077907823,0.0533846281,-0.0197199378,-
0.0196586996],[-0.0110050291,-0.0242669974,0.0505006835,0.0275756605,0.105092898,-0.0376476645,-
0.0762326866,0.0276804604],[-0.0976924598,0.0391854085,-0.0141086681,0.0269201249,-0.0526741482,-
0.0203541424,-0.0667346716,-0.0057867416],[-0.0809081569,-0.0117225731,-0.0270767137,-
0.0074305944,-0.0496960655,0.0531772338,-0.0928004384,0.0294681992],[-0.0648343191,0.033810243,-
0.0313544422,-0.0063377586,0.0217672195,0.0425700881,-0.0685876012,-0.037825048],[-0.1137577444,-
0.0154528851,-0.0387987643,0.0237913579,-0.0190423597,-0.0365628861,-0.0272838157,0.0166711546],[-
0.0282623619,0.0022903634,-0.0708298087,-0.0558150895,-0.0013412465,-0.0244858544,-0.0343662649,-
0.0135337785],[-0.1104128957,0.0239975192,-0.0244038738,-0.0074469303,-0.0177662577,0.0057930686,-
0.0428019688,-0.0244289562],[-0.0451579466,0.0520865507,-0.0094025219,0.0118894158,-
0.0648013577,-0.0122414632,0.0000643103,-0.0415868312],[-0.0886026621,0.0346491151,-0.0936349779,-
0.0328464732,-0.0596412234,0.0380754471,-0.0481899679,0.0351417623],[-0.062903963,0.0366867296,-
0.0671236813,-0.0331896245,0.00912847,-0.0117816087,-0.0623806491,-0.0489633642],[-0.1062339991,-
0.0182165653,-0.0384030677,-0.049000904,-0.0297394134,0.0507741459,-0.0590156689,0.0042186659],[-
0.0655230284,-0.0152729163,-0.040764153,0.0108577851,0.0204557627,0.0553223975,-0.0098171271,-
0.0185332336],[-0.086137265,0.0079337712,-0.0300996657,0.0136304209,0.0023132216,-
0.0276144557,0.0024335615,0.0041935169],[-0.0926613957,0.0348509587,-0.0282577146,-
0.0021309936,0.0195951276,0.0242319237,-0.0867084935,-0.0303266309],[-0.0527052283,0.0738844797,-
0.0327605456,0.0233741403,0.0127363019,0.0397874378,-0.0238321126,-0.0234394111],[-
0.0875016078,0.0483934507,-0.0637599453,0.0238931421,-0.0292015113,-0.0354836248,-0.0078475857,-
0.0428684168],[-0.0968683437,0.023401808,-0.0867186263,-0.0019967486,-0.043619439,0.0408246256,-
0.0914459601,-0.0269118957],[-0.0314008556,0.0583638065,-0.0629426241,0.0462925099,-
0.0173312239,0.0523343422,-0.0680754483,-0.0483582541],[-0.1000395641,0.0204116944,-
0.0048646885,0.0533132404,-0.0013003877,0.020463245,-0.091444172,-0.0125555154],[-
0.0903037712,0.0633033514,-0.0521341041,0.0165143125,-0.067955777,0.0549336262,-0.0217242986,-
0.0051297308],[-0.0463354439,-0.001126894,-0.0578635745,-0.0226509105,-0.0180652086,0.0319400653,-
0.0667770281,-0.0421370417],[-0.0796041787,0.0605712049,-0.030303454,0.0321338288,-0.0651268736,-
0.0003121346,-0.0893633217,-0.0459697694],[-0.0253375992,0.0586420111,-0.083028771,-

0.0312443115,-0.0419821627,-0.0239030756,-0.0432118811,-0.0545598008],[0.1129746288,-
0.0144239282,0.0394194089,0.0450047478,0.0239712521,-0.0041741244,0.056005016,-0.0341856368],[-
0.0879750699,0.0274951402,-0.0778044164,-0.0063563818,0.0221106634,-0.0227532648,-
0.0337350629,-0.0577941015],[0.0052057616,-0.0076587838,-0.0357088856,0.0170469936,-
0.0300128311,-0.0045791264,0.0240043458,-0.0538814738],[-0.0435640849,0.0551614799,-
0.0377208367,0.0239589661,0.0153081194,-0.0299693178,-0.0078395531,-0.0172580741],[-
0.089750737,0.002687284,-0.0327109061,-0.0190543588,-0.0183317158,-0.0186330117,-0.0300050508,-
0.0049766293],[-0.0617307387,0.0570785664,-0.0957253426,-0.0419555083,-0.0404380448,-
0.0296482742,-0.0400823094,-0.052381631],[-0.0028140291,-0.0363146253,-0.0830766857,0.0459738672,-
0.0008706906,0.0482100584,0.0369101912,-0.0254053846],[-0.1060718447,0.0533535294,-0.0124383941,-
0.0481052585,-0.0159467235,0.0437121503,-0.0373585932,0.0067033148],[-0.0716029555,0.0494868793,-
0.02825954,0.0206357539,-0.0450302809,0.0430571102,-0.0696179047,0.0003198251],[-
0.0339473598,-0.0198790804,-0.0145128313,-0.0321352817,-0.0014832687,-0.0332573541,-
0.0706539527,0.0241565295],[-0.041627232,0.0105764754,-0.0028503521,-0.0275784023,-
0.0169867314,-0.0152813829,0.0311609209,-0.0055409656],[-0.0560917333,0.0728324056,-
0.086179018,-0.0263916943,-0.0134663749,0.0310398471,-0.0845057219,0.0376214646],[0.0913455263,-
0.0244095344,0.04827068,0.0233552195,-0.008574103,-0.0283995494,0.0031798296,-0.0317710489],[-
0.0624284483,-0.0016197762,-0.0785579458,-0.0338378102,0.000383949,0.0109281447,-
0.041599106,0.0070661204],[-0.0086845122,-0.017280519,-0.0152701437,0.0125841666,0.0271950159,-
0.0383302346,0.0036404752,0.0427528396],[0.0793109238,-0.0227750819,-0.010209349,0.0119409207,-
0.0098429937,-0.0377885513,0.0298203342,-0.0102348486],[-0.0240933504,0.0260769036,-
0.0036114398,-0.0170906186,-0.0000439026,-0.0470194109,0.0108775385,0.0033790595],[0.0500849932,-
0.0341464207,0.0060232393,0.0181315113,0.0037009821,0.0123735731,0.0424274057,-
0.0054885577],[-0.058255434,-0.0118014067,-0.0580486469,-0.0455605276,-0.0682392865,-
0.0163716339,0.0017862685,0.0341342129],[-0.0226843059,0.0439862236,-0.0259097423,-
0.0413032919,0.0158601198,-0.0106990431,-0.0477556884,0.0029960452],[-0.0385067165,0.0142230922,-
0.0072720801,0.0303163808,0.0142978448,-0.0209562685,-0.0893662795,-0.038411539],[-
0.0713111758,0.0383256078,-0.0761647969,0.0015004368,-0.0180589538,0.0243014079,-0.0387510918,-
0.0375607759],[-0.1070147157,-0.0216446109,-0.0253466871,-0.0559930541,-0.0718536079,0.052701626,-
0.0909493491,0.0051109158],[-0.0939891934,0.0235079378,-0.0350640751,-0.0262695979,-
0.0056846449,0.0010250245,-0.0683567822,0.022541048],[-0.1046996713,0.0637534857,-0.0618929118,-
0.0553447343,-0.0366071835,0.0017380636,-0.0060565625,-0.0285031348],[0.0267615207,-
0.0164158121,0.0243424661,0.0221200697,0.041919861,0.0484827645,-0.0045593604,0.0202077739],[-
0.0672270656,0.006625135,-0.0738261864,0.0167296026,0.0209824499,0.0530312769,-0.0255036,-
0.0328175128],[0.0385857783,0.0345200114,0.012945978,-0.0366297849,0.0055175275,-
0.0025519063,0.0558558255,0.0481611751],[-0.0206962395,0.000667054,0.0023179185,0.0031977177,0.0110240057,-
0.0364335552,0.0004348502,-0.0018964445],[-0.1055001467,-0.0173251275,-0.0073329764,-
0.0535804629,-0.0067843311,0.0287012067,-0.0517770648,-0.041283831],[-0.0876494646,0.0599330775,-
0.0007948491,0.0177025273,0.0052934354,0.0361828543,-0.0200640485,-0.0020867223],[-
0.0504096933,-0.0046923552,-0.0253490042,0.0042371131,-0.0158475023,0.0011175644,-
0.0807759166,0.0389074348],[-0.0365884416,0.049988497,-0.034362331,-0.026570566,-
0.0239959899,0.03372458,0.0029976598,-0.014007993],[-0.0184149221,0.0066243359,-
0.01104559,0.0150875505,0.010089457,-0.0374504104,-0.0416944511,-0.0138070658],[0.0306735747,-
0.0084939497,-0.0207424723,0.0112914825,0.0215460621,-0.0131431986,0.0598166697,-
0.0047641485],[-0.0131553896,0.026114041,-0.0776950493,-0.0369460955,0.0089351963,0.0063564447,-
0.0250268783,-0.0383680686],[0.0264446568,0.0177899543,0.0337854289,0.0486565344,0.0455680899,-
0.0133411037,-0.0472401679,-0.0042054271],[-0.0156883281,0.0023868394,-0.0359831229,-
0.0533350222,-0.0620505922,0.0117895743,-0.0465891063,0.014333277],[0.0037823613,-
0.0485859625,0.0304957367,0.0142664108,0.0231500044,0.0199270826,-0.0771579742,0.0058415439],[0.0093906112,0.043941568
0.0403797813,0.0201416481,-0.0099902553,-0.0186526831,-0.053171806,0.0428802408]],[0.0652318001,-
0.027705403,0.0473561734,0.0063774441,0.0273798443,-0.0068882219,0.0441013537,0.0095654931],[[0.0664635003,-
0.0361788496,0.0981992334,-0.0049552293,0.0115166651,0.0670768172],[-0.0061368346,0.0072507635,-
0.0050328653,0.0159201734,0.0065031671,0.026376985],[0.0508560166,-0.0063855532,0.1221936271,-
0.0424587801,-0.0302463807,0.0992918313],[0.1041004956,-0.0349653065,0.0151295438,-

0.0530779213,-0.0269837249,0.0135957841],[0.1047551259,-0.0295952559,0.0221561752,-0.0217163917,-0.0311993714,0.0422756672],[-0.0463010445,0.032602299,-0.0292981397,0.0043135858,0.0399170071,-0.0342914276],[0.0540453047,-0.0274718888,0.0625797585,0.0309408139,-0.0458704196,0.0513330922],[-0.0057910825,-0.008349441,0.0426463857,-0.0266905036,0.0196692776,0.0481408983]],[0.0441856496,-0.0279886983,0.0708117858,-0.0080904942,-0.0135130016,0.0418888181],[[-0.1451997608],[-0.0171129871],[-0.1398460418],[-0.0024810787],[0.024602361],[-0.0444240533]],[-0.0354687795]]'}, 'acc_image_file': '/media/sf_shared/accuracy_job_16_result_12.png', 'created': '2018-02-15 19:11:06', 'updated': '2018-02-15 19:11:06', 'deleted': ''}}

Get the Deep Neural Network Accuracy, JSON and Weights

This will display all the recent training runs in a list sorted by newest.

```
/opt/antinex/api/tests/get-recent-results.py
```

Here's the training node in the list from the run above (yours will look a little different):

```
{
    "acc_data": {
        "accuracy": 87.4074073926902
    },
    "acc_image_file": "/media/sf_shared/accuracy_job_16_result_12.png",
    "created": "2018-02-15 19:11:06",
    "deleted": "",
    "error_data": null,
    "id": 12,
    "job_id": 16,
    "model_json": "{\"class_name\": \"Sequential\", \"config\": [{\"class_name\": \
↪"Dense\", \"config\": {\"name\": \"dense_4\", \"trainable\": true, \"batch_input_
↪shape\": [null, 68], \"dtype\": \"float32\", \"units\": 8, \"activation\": \"relu\",
↪ \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \"RandomUniform\", \
↪"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}, \"bias_
↪initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_regularizer\":␣
↪null, \"bias_regularizer\": null, \"activity_regularizer\": null, \"kernel_
↪constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\", \
↪"config\": {\"name\": \"dense_5\", \"trainable\": true, \"units\": 6, \"activation\
↪": \"relu\", \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \
↪"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}
↪, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_
↪regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\": null, \
↪"kernel_constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\",
↪ \"config\": {\"name\": \"dense_6\", \"trainable\": true, \"units\": 1, \
↪"activation\": \"sigmoid\", \"use_bias\": true, \"kernel_initializer\": {\"class_
↪name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\
↪": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \
↪"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\":␣
↪null, \"kernel_constraint\": null, \"bias_constraint\": null}}], \"keras_version\":␣
↪\"2.1.4\", \"backend\": \"tensorflow\"}",
    "model_weights": {
        "weights": "[[[0.052631028,-0.0195708107,0.0235880036,-0.0160469897,0.
↪0599223375,-0.019402137,-0.0315921232,-0.0252120867],[-0.0266926326,-0.0446610935,0.
↪0770860612,-0.0265123285,-0.0030328943,-0.0082395915,-0.0333385319,-0.0549867488],[-
↪0.0178412981,-0.0179316401,-0.0041324655,0.0311650522,-0.0275568571,-0.0177768506,-
↪0.0017130028,0.0179622211],[-0.0210418832,0.0191573389,-0.0079981312,-0.0045636012,-
↪0.0231961794,-0.0242097769,-0.0480531305,0.0070477622],[-0.0076428168,-0.0474744439,
↪0.0809031352,0.0383856446,0.077907823,0.0533846281,-0.0197199378,-0.0196586996],[-0.
↪0110050291,-0.0242669974,0.0505006835,0.0275756605,0.105092898,-0.0376476645,-0.
↪0762326866,0.0276804604],[-0.0976924598,0.0391854085,-0.0141086681,0.0269201249,-0.
↪0526741482,-0.0203541424,-0.0667346716,-0.0057867416],[-0.0809081569,-0.0117225731,-
↪0.0270767137,-0.0074305944,-0.0496960655,0.0531772338,-0.0928004384,0.0294681992],[-
↪0.0648343191,0.033810243,-0.0313544422,-0.0063377586,0.0217672195,0.0425700881,-0.
↪0685876012,-0.037825048],[-0.1137577444,-0.0154528851,-0.0387987643,0.0237913579,-0.
↪0190423597,-0.0365628861,-0.0272838157,0.0166711546],[-0.0282623619,0.0022903634,-0.
↪0708298087,-0.0558150895,-0.0013412465,-0.0244858544,-0.0343662649,-0.0135337785],[-
```

(continues on next page)

(continued from previous page)

```
    }
}
```

4.4.4 Protecting Spring with a Deep Neural Network

This guide is a walkthrough for preparing and training a deep neural network for defending Spring application servers. The accuracy is currently **66%** without tuning the DNN or adding in actual exploits or sql-injection attacks into the attack datasets. Please note the `non-attack` training data is recorded from a multi-user simulation against a Django application server. Sorry I have not had enough free time to create a true Spring non-attack dataset (PRs welcome though!).

In the future I am looking to extend the full datasets to include the TCP payload data stream (hex bytes) for sentiment analysis using an embedding Keras layer (https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html). I imagine deserialized payloads will only increase the default accuracy, but it is only an assumption for now.

Setup

1. Run these commands to clone the repositories to the same directories for making debugging easier for all users.

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /
↪opt/antinex/api
git clone https://github.com/jay-johnson/network-pipeline-datasets.git /opt/
↪antinex/datasets
git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/
↪antinex-datasets
```

2. Start the REST API

If the REST API is not running, please start it in a new terminal so it can process the prepare and training requests.

```
cd /opt/antinex/api
source ~/.venvs/venvdrfpipeline/bin/activate
./install.sh
./start.sh
```

(Optional) Prepare Attack Dataset

If you want to prepare your own attack dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/spring/configs/spring-
↪attack-prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Check the files were updated:

```
ls -l /opt/antinex/antinex-datasets/v1/webapps/spring/inputs/attack/
total 13772
-rw-rw-r-- 1 jay jay     2080 Feb 15 11:18 cleaned_v1_spring_attack_metadata.json
```

(continues on next page)

(continued from previous page)

```
-rw-rw-r-- 1 jay jay     2451 Feb 15 11:18 fulldata_v1_spring_attack_metadata.json
-rw-rw-r-- 1 jay jay  1356260 Feb 15 11:18 v1_spring_cleaned_attack.csv
-rw-rw-r-- 1 jay jay 12731474 Feb 15 11:18 v1_spring_full_attack.csv
```

(Optional) Prepare Full Dataset

If you want to prepare your own full dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/spring/configs/spring-
↪prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Confirm Dataset is Ready

```
/opt/antinex/antinex-datasets/tools/describe-v1-training.py /opt/antinex/antinex-
↪datasets/v1/webapps/spring/training-ready/v1_spring_cleaned.csv
```

Hopefully your dataset has both attack and non-attack records like:

```
2018-02-15 11:19:42,038 - describe-training-data - INFO - total records=32000␣
↪attack=10800 nonattack=21200 percent_attack=33.75% percent_nonattack=66.25%
```

What you don't want to see is this in the output:

```
2018-02-15 08:47:41,389 - describe-training-data - INFO - total records=21200␣
↪attack=0 nonattack=21200 percent_attack=0.00% percent_nonattack=100.00%
```

That means the prepare step failed to add the attack data into the dataset correctly. Please go back to the `Prepare Dataset` step and review paths to the files are correct.

Train Dataset

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/spring/configs/spring-train-
↪v1.json
/opt/antinex/api/tests/create-keras-dnn.py
```

From the logs taken during creation of this doc, the model is 66% accurate at predicting attack records.

/opt/antinex/api/tests/create-keras-dnn.py INFO:create-keras-dnn:Logging in user url=http://localhost:8010/api-token-auth/ INFO:create-keras-dnn:logged in user=root token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MTg3MjI3MDIsIm INFO:create-keras-dnn:building post data INFO:create-keras-dnn:Running ML Job url=http://localhost:8010/ml/ test_data={'csv_file': '/opt/antinex/antinex-datasets/v1/webapps/spring/training-ready/v1_spring_cleaned.csv', 'meta_file': '/opt/antinex/antinex-datasets/v1/webapps/spring/training-ready/cleaned_v1_spring_metadata.json', 'title': 'Spring - Keras DNN - Dataset v1', 'desc': 'Training Spring DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'predict_feature': 'label_value', 'training_data': '{}', 'pre_proc': '{}', 'post_proc': '{}', 'meta_data': '{}', 'version': 1} INFO:create-keras-dnn:SUCCESS - Post Response status=201 reason=Created INFO:create-keras-dnn:{'job': {'id': 17, 'user_id': 1, 'user_name': 'root', 'title': 'Spring - Keras DNN - Dataset v1', 'desc': 'Training Spring DNN using Attack and Non-attack data captured using the network-pipeline',

‘ds_name’: ‘cleaned’, ‘algo_name’: ‘dnn’, ‘ml_type’: ‘keras’, ‘status’: ‘initial’, ‘control_state’: ‘active’, ‘predict_feature’: ‘label_value’, ‘training_data’: {}, ‘pre_proc’: {}, ‘post_proc’: {}, ‘meta_data’: {}, ‘tracking_id’: ‘ml_00885343-59fa-4259-91e3-0ae6b8a715cb’, ‘version’: 1, ‘created’: ‘2018-02-15 19:20:02’, ‘updated’: ‘2018-02-15 19:20:02’, ‘deleted’: ‘’}, ‘results’: {‘id’: 13, ‘user_id’: 1, ‘user_name’: ‘root’, ‘job_id’: 17, ‘status’: ‘finished’, ‘version’: 1, ‘acc_data’: {**‘accuracy’: 66.09375**}, ‘error_data’: None, ‘model_json’: ‘{“class_name”: “Sequential”, “config”: [{“class_name”: “Dense”, “config”: {“name”: “dense_1”, “trainable”: true, “batch_input_shape”: [null, 68], “dtype”: “float32”, “units”: 8, “activation”: “relu”, “use_bias”: true, “kernel_initializer”: {“class_name”: “RandomUniform”, “config”: {“minval”: -0.05, “maxval”: 0.05, “seed”: null}}, “bias_initializer”: {“class_name”: “Zeros”, “config”: {}}, “kernel_regularizer”: null, “bias_regularizer”: null, “activity_regularizer”: null, “kernel_constraint”: null, “bias_constraint”: null}}, {“class_name”: “Dense”, “config”: {“name”: “dense_2”, “trainable”: true, “units”: 6, “activation”: “relu”, “use_bias”: true, “kernel_initializer”: {“class_name”: “RandomUniform”, “config”: {“minval”: -0.05, “maxval”: 0.05, “seed”: null}}, “bias_initializer”: {“class_name”: “Zeros”, “config”: {}}, “kernel_regularizer”: null, “bias_regularizer”: null, “activity_regularizer”: null, “kernel_constraint”: null, “bias_constraint”: null}}, {“class_name”: “Dense”, “config”: {“name”: “dense_3”, “trainable”: true, “units”: 1, “activation”: “sigmoid”, “use_bias”: true, “kernel_initializer”: {“class_name”: “RandomUniform”, “config”: {“minval”: -0.05, “maxval”: 0.05, “seed”: null}}, “bias_initializer”: {“class_name”: “Zeros”, “config”: {}}, “kernel_regularizer”: null, “bias_regularizer”: null, “activity_regularizer”: null, “kernel_constraint”: null, “bias_constraint”: null}}], “keras_version”: “2.1.4”, “backend”: “tensorflow”}’, ‘model_weights’: {‘weights’: ‘[[[-0.0173000526,-0.0064027878,-
0.0169708095,0.0157266911,0.0662872419,-0.0328218415,0.1295523047,0.0344657972],[-0.0088931685,-
0.0385553166,0.0037408469,0.0977382362,0.0767394751,-0.05592921,0.0982304141,0.014463977],[-
0.0322886854,-0.0368546396,0.0107642207,-0.0718934014,-0.0221321229,0.0071108998,-
0.1037670299,-0.0087477071],[0.0416910201,-0.0537477769,0.019645201,0.0635252744,0.0881028324,-
0.0220302157,0.0654686466,0.012742796],[-0.0213366691,-0.0323441252,-0.0488818437,0.087537989,0.0398393236,-
0.0530910976,0.1179806888,-0.0108120786],[-0.0286345202,-0.0170552637,-0.01019214,0.096555151,0.0748246536,-
0.0755968541,0.0934303999,-0.0308071431],[0.0030201294,-0.0528637655,-0.0346440263,-0.0669601113,-
0.0656459033,0.0276927054,-0.0825652406,-0.0032823205],[-0.0056472942,-0.0407883152,0.0274266116,-
0.0543933474,-0.0784520358,-0.0397142395,-0.1286949962,0.034545105],[0.0393268168,-
0.0346527621,0.0186092779,-0.034386944,-0.0663760602,-0.0158903264,-0.0826205313,0.0005784247],[-
0.0380487517,-0.0278219841,-0.0147631671,-0.016108444,-0.0466402136,0.0435810089,-0.1015856043,-
0.0084425164],[-0.0488428921,-0.0293625016,0.026683338,-0.0435961001,-0.1025879383,-0.0353616923,-
0.0771325007,0.0552083217],[-0.0025817794,-0.0247298032,0.0419541076,-0.0949794203,-
0.026031835,0.0116581451,-0.1078904942,0.0487319119],[-0.0153548149,0.0402287059,-0.002908526,-
0.0893430561,-0.0536826178,-0.0266579315,-0.1445246637,-0.0359159745],[0.0435531884,-
0.025654668,-0.038579829,-0.0826740116,-0.0893285349,0.0105355503,-0.066000931,-0.034613315],[-
0.0488474704,-0.0199130196,0.0422015861,-0.0321295559,-0.0455789417,0.0059766336,-
0.0860127956,0.0176941063],[0.0074663856,0.0338624604,-0.0018213085,-0.0542690866,-0.0301753003,-
0.0390444547,-0.0836630166,0.0284509044],[-0.0511877276,-0.0219058525,-0.0340818726,-0.0345237553,-
0.0381731167,-0.0017185912,-0.1276020557,0.0149726318],[0.0081929648,0.0145618366,0.0185622554,-
0.1079799682,-0.0896312669,-0.0002979506,-0.098251991,0.0011839687],[-0.0210764948,0.0120229824,0.0251360014,-
0.0380748846,-0.0478647351,-0.0444115773,-0.0653204471,-0.0378449708],[-0.0005556387,-
0.0434047244,0.0310283601,-0.0910413787,-0.0988587886,-0.0266462117,-0.0923515782,-0.0181508325],[-
0.0096717393,0.0347021855,0.018473519,-0.0408929698,-0.0938989446,0.0198299177,-0.0886605158,-
0.0234003849],[0.0162979532,-0.0537309386,-0.0134870214,-0.0631217659,-0.0191594698,0.0320905186,-
0.1432752609,0.0562950373],[-0.0522111617,-0.0289670061,-0.0153701846,-0.0334702469,-0.0374529734,-
0.0267687216,-0.0723697096,0.0163344964],[-0.0087670712,-0.0421352983,0.0385387503,-0.1075451523,-
0.0103809508,-0.0249420628,-0.12667723,-0.0193419773],[-0.0398854949,0.0253311843,0.0518782474,-
0.0364522263,-0.0960612893,-0.0428358912,-0.0894641876,0.0275488924],[0.0149986902,-
0.021600645,0.0153730037,-0.0267963409,-0.098190546,0.0403022617,-0.0808125436,0.0382996574],[-
0.0023344536,-0.0428262725,0.0202057045,-0.0766336545,-0.0849173516,0.0463943556,-
0.1428285539,0.0519381054],[0.042711705,-0.050877668,-0.0172832552,-0.0740391836,-
0.0718216076,-0.0103346519,-0.0579395629,0.0442523919],[0.0060770563,0.0171900522,-
0.0219493955,0.1135827675,0.1139239743,0.0393700376,0.1700653881,-0.0374217182],[0.0302413236,0.0040957471,0.044197157
0.0525682308,-0.0123297134,-0.0051418832,-0.1161182448,0.0289911013],[0.0080568166,-

0.0293921139,0.0368611924,-0.031907361,-0.0080400519,0.0517797805,0.0480450913,0.0234247819],[0.0165923182,-
0.0182517283,-0.0100761745,-0.1003762558,-0.0173195116,-0.0350859612,-0.085473381,-
0.0150983492],[0.0067120269,-0.053849794,0.0544353202,-0.0249461662,-0.0422286354,0.0218810663,-
0.1306264699,-0.0048307315],[0.0156460945,0.0277566575,-0.024820514,-0.1039614528,-
0.0260938685,-0.0021631087,-0.0492782556,0.0043323012],[0.0227578375,0.0221582968,0.0007841409,-
0.0886592641,-0.0884859934,0.0525981635,-0.0798395574,0.0458145142],[0.0236985628,-
0.0366411731,0.0548207089,-0.0146643622,-0.0687453374,-0.0444763452,-0.1426427066,-
0.03775746],[-0.0252918322,0.0014541645,0.0016110033,-0.0387289189,-0.0257748235,-
0.0430029631,-0.1233881786,0.0235356223],[0.0122827264,0.0251069088,-0.0027357663,-
0.1024575979,-0.0682099089,-0.0049612666,-0.0609179363,0.0038837341],[-0.0354516096,-
0.033333566,0.0563913882,-0.0726352409,-0.0318466015,0.037089549,-0.074870795,0.0281760283],[-
0.0080876146,-0.0468189716,0.0424663126,-0.0533910729,-0.0204824172,0.0296455175,-
0.0709483698,0.0496929996],[-0.0302802306,-0.0335655995,-0.0102529833,0.1016696915,0.0252007376,-
0.0023698036,0.0479990542,-0.0358106755],[-0.0043822778,-0.0191927347,0.0084026409,-
0.0475930236,-0.0715683997,-0.0273918658,-0.1038119346,0.0135136517],[0.0292350873,-
0.0316817425,0.0423877165,0.0283667725,-0.0179847665,-0.0016444682,-0.0026728681,0.0337618776],[-
0.037272051,-0.0279664211,0.0315952115,0.0440362394,0.0267583579,0.0005767916,0.0571529903,0.0346558914],[0.0195410363
0.0069632046,-0.0175475031,-0.0679977313,-0.0776399076,-0.0129115684,-0.1174754798,-0.0326450057],[-
0.0015634091,0.0309963748,0.0030699954,0.0250998698,-0.0931123495,-0.0225014891,-0.0774686038,-
0.0475768745],[0.0129500544,0.0356599353,-0.0125144441,-0.0188169945,-0.0661799088,0.0368166193,-
0.1156292632,0.0554054491],[0.0062011112,0.0128063438,-0.0244099833,-0.0727104247,-0.0146835009,-
0.0259854402,-0.0947751999,0.011442353],[0.0190492067,-0.0187272225,0.0457368046,-0.0230206568,-
0.1026833132,-0.0111416653,-0.0815310627,0.0057581617],[0.0429183543,0.0011941099,0.0008057182,-
0.061610911,-0.0891352072,0.0450408459,-0.0828220621,0.0327208899],[-0.0100156097,-
0.0446096547,0.0161210727,-0.0211098623,-0.0092519093,-0.0104807913,-0.0842147619,0.0085202316],[-
0.0015388664,0.0369447805,-0.0316719003,-0.0160452444,-0.0185811594,0.022215724,-
0.1362029761,0.0196939111],[-0.0043828255,-0.0051892484,-0.0405620411,-0.020046562,-
0.0957702845,-0.0233690925,-0.0848989114,0.0541716181],[0.0280559696,0.0005700476,-
0.0328265429,0.0220364444,0.030539047,-0.0166563876,0.0155522265,-0.0200448763],[0.0233413521,-
0.0373235866,-0.0206750352,-0.0256250873,-0.0491497479,0.0132879745,-0.100703612,0.03170399],[0.0024619398,0.0280608162
0.0210946053,0.0260587037,0.0045587812,0.0193805881,-0.1318970025,0.0579834767],[-0.0345870592,-
0.0012203066,-0.0320372172,-0.0359899588,-0.0393892862,-0.0036178678,-0.0803916231,-
0.0077602961],[-0.0385007709,0.0408850051,0.0327539444,-0.063881211,-0.0336246081,-
0.0460557714,-0.0942867622,0.0072741951],[-0.0031206983,0.0007216324,0.0119840298,-
0.097627461,-0.088842459,0.0227372032,-0.115960598,0.0427624248],[0.0174984057,0.0085226558,-
0.0363096781,-0.0492029749,-0.0409341604,-0.0284969937,-0.0751886219,0.0110676056],[-
0.0016668823,0.0221300963,0.0554900318,-0.0871766955,-0.0556310974,-0.0455159545,-0.063777566,-
0.0322403684],[0.031324394,0.0298806243,0.0565144718,-0.0162632931,-0.0474709682,0.0033299716,-
0.1447925121,0.0057630157],[0.0300826635,0.0100916857,0.0247726478,0.0075516687,-0.0427784547,-
0.0050428738,-0.0933085829,0.0140131107],[-0.0086429873,-0.0454958193,0.0346922912,0.026505664,-
0.0862411782,0.0473857149,-0.0348779708,-0.0183145478],[-0.0128766438,0.0098509705,-
0.040062733,0.031110106,-0.0421553552,-0.0158805288,-0.0655305088,-0.0007482105],[-0.0449128598,-
0.0197112951,0.0278943479,-0.0182558633,-0.0181404762,-0.0328727961,-0.0650151819,0.012640168],[-
0.0107919117,0.0400995612,-0.0031944313,0.021298036,-0.0168119706,-0.0131717259,-
0.1278857887,0.0147369802],[-0.0134499101,0.0214591976,0.0350737795,0.0236329325,0.0209473409,0.0107630836,-
0.1442049593,0.0037561236]],[0.0035855921,0.0059448453,-0.008031507,0.0608890243,0.0532443672,-
0.0005787634,0.0981881544,-0.0074393484],[[0.0541371442,-0.0233311709,0.03808631,-0.0074524796,-
0.0161964875,-0.0531380847],[0.0440565683,0.0084736859,0.020711517,0.0032913573,-0.0160148256,-
0.0212113783],[-0.0339465365,-0.0085006962,0.0432360955,0.0263918042,0.0323373266,0.0352469683],[0.0598729029,-
0.0117976377,0.030797217,0.0513003804,0.0737307891,-0.0350890942],[0.0600481182,0.0427516364,-
0.0121414801,0.0573144294,0.0142900757,0.0204450823],[0.0106062582,-0.024731813,0.0710651278,0.0536883213,0.0389724039
0.013540511],[0.0798899829,0.0936565548,0.1172549501,0.0896263868,0.1345335245,-
0.0165757835],[0.0281711705,0.039581731,0.0377913043,0.0001205466,-0.0142628271,0.0254401863]],[0.0620664246,0.03224653
0.0073743802],[[-0.0597119369],[-0.0197188053],[-0.0170194674],[-0.1285354942],[-

0.0675625801],[0.0375708863]],[-0.0585541092]]'}, 'acc_image_file': '/media/sf_shared/accuracy_job_17_result_13.png', 'created': '2018-02-15 19:21:28', 'updated': '2018-02-15 19:21:28', 'deleted': ''}}

Get the Deep Neural Network Accuracy, JSON and Weights

This will display all the recent training runs in a list sorted by newest.

```
/opt/antinex/api/tests/get-recent-results.py
```

Here's the training node in the list from the run above (yours will look a little different):

```
{
    "acc_data": {
        "accuracy": 66.09375
    },
    "acc_image_file": "/media/sf_shared/accuracy_job_17_result_13.png",
    "created": "2018-02-15 19:21:28",
    "deleted": "",
    "error_data": null,
    "id": 13,
    "job_id": 17,
    "model_json": "{\"class_name\": \"Sequential\", \"config\": [{\"class_name\": \
↪"Dense\", \"config\": {\"name\": \"dense_1\", \"trainable\": true, \"batch_input_
↪shape\": [null, 68], \"dtype\": \"float32\", \"units\": 8, \"activation\": \"relu\",
↪ \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \"RandomUniform\", \
↪"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}, \"bias_
↪initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_regularizer\":␣
↪null, \"bias_regularizer\": null, \"activity_regularizer\": null, \"kernel_
↪constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\", \
↪"config\": {\"name\": \"dense_2\", \"trainable\": true, \"units\": 6, \"activation\
↪": \"relu\", \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \
↪"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}
↪, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_
↪regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\": null, \
↪"kernel_constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\",
↪ \"config\": {\"name\": \"dense_3\", \"trainable\": true, \"units\": 1, \
↪"activation\": \"sigmoid\", \"use_bias\": true, \"kernel_initializer\": {\"class_
↪name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\
↪": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \
↪"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\":␣
↪null, \"kernel_constraint\": null, \"bias_constraint\": null}}], \"keras_version\":␣
↪\"2.1.4\", \"backend\": \"tensorflow\"}",
    "model_weights": {
        "weights": "[[[-0.0173000526,-0.0064027878,-0.0169708095,0.0157266911,0.
↪0662872419,-0.0328218415,0.1295523047,0.0344657972],[-0.0088931685,-0.0385553166,0.
↪0037408469,0.0977382362,0.0767394751,-0.05592921,0.0982304141,0.014463977],[-0.
↪0322886854,-0.0368546396,0.0107642207,-0.0718934014,-0.0221321229,0.0071108998,-0.
↪1037670299,-0.0087477071],[0.0416910201,-0.0537477769,0.019645201,0.0635252744,0.
↪0881028324,-0.0220302157,0.0654686466,0.012742796],[-0.0213366691,-0.0323441252,-0.
↪0488818437,0.087537989,0.0398393236,-0.0530910976,0.1179806888,-0.0108120786],[-0.
↪0286345202,-0.0170552637,-0.01019214,0.096555151,0.0748246536,-0.0755968541,0.
↪0934303999,-0.0308071431],[0.0030201294,-0.0528637655,-0.0346440263,-0.0669601113,-
↪0.0656459033,0.0276927054,-0.0825652406,-0.0032823205],[-0.0056472942,-0.0407883152,
↪0.0274266116,-0.0543933474,-0.0784520358,-0.0397142395,-0.1286949962,0.034545105],
↪[0.0393268168,-0.0346527621,0.0186092779,-0.034386944,-0.0663760602,-0.0158903264,-
↪0.0826205313,0.0005784247],[-0.0380487517,-0.0278219841,-0.0147631671,-0.016108444,-
↪0.0466402136,0.0435810089,-0.1015856043,-0.0084425164],[-0.0488428921,-0.0293625016,
↪0.026683338,-0.0435961001,-0.1025879383,-0.0353616923,-0.0771325007,0.0552083217],[-
↪0.0025817794,-0.0247298032,0.0419541076,-0.0949794203,-0.026031835,0.0116581451,-0.
↪1078904942,0.0487319119],[-0.0153548149,0.0402287059,-0.002908526,-0.0893430561,-0.
↪0536826178,-0.0266579315,-0.1445246637,-0.0359159745],[0.04355[illegible]
↪038579829,-0.0826740116,-0.0893285349,0.0105355503,-0.066000931,-0.034613315],[-0.
↪0488474704,-0.0199130196,0.0422015861,-0.0321295559,-0.0455789417,0.0059766336,-0.
↪0860127956,0.0176941063],[0.0074663856,0.0338624604,-0.0018213085,-0.0542690866,-0.
↪0301753003,-0.0390444547,-0.0836630166,0.0284509044],[-0.0511877276,-0.0219058525,-
```

(continues on next page)

(continued from previous page)

```
    },
    "status": "finished",
    "updated": "2018-02-15 19:21:28",
    "user_id": 1,
    "user_name": "root",
    "version": 1
}
```

4.4.5 Protecting Vue with a Deep Neural Network

This guide is a walkthrough for preparing and training a deep neural network for defending Vue application servers. The accuracy is currently **83%** without tuning the DNN or adding in actual exploits or sql-injection attacks into the attack datasets. Please note the `non-attack` training data is recorded from a multi-user simulation against a Django application server. Sorry I have not had enough free time to create a true Vue non-attack dataset (PRs welcome though!).

In the future I am looking to extend the full datasets to include the TCP payload data stream (hex bytes) for sentiment analysis using an embedding Keras layer (https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html). I imagine deserialized payloads will only increase the default accuracy, but it is only an assumption for now.

Setup

1. Run these commands to clone the repositories to the same directories for making debugging easier for all users.

   ```
   mkdir -p -m 777 /opt/antinex
   git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /
   ↪opt/antinex/api
   git clone https://github.com/jay-johnson/network-pipeline-datasets.git /opt/
   ↪antinex/datasets
   git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/
   ↪antinex-datasets
   ```

2. Start the REST API

   If the REST API is not running, please start it in a new terminal so it can process the prepare and training requests.

   ```
   cd /opt/antinex/api
   source ~/.venvs/venvdrfpipeline/bin/activate
   ./install.sh
   ./start.sh
   ```

(Optional) Prepare Attack Dataset

If you want to prepare your own attack dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/vue/configs/vue-attack-
↪prepare-v1.json
/opt/antinex/api/tests/build-new-dataset.py
```

Check the files were updated:

```
ls -l /opt/antinex/antinex-datasets/v1/webapps/vue/inputs/attack/
total 5752
-rw-rw-r-- 1 jay jay    2077 Feb 15 11:28 cleaned_v1_vue_attack_metadata.json
-rw-rw-r-- 1 jay jay    2408 Feb 15 11:28 fulldata_v1_vue_attack_metadata.json
-rw-rw-r-- 1 jay jay  553131 Feb 15 11:28 v1_vue_cleaned_attack.csv
-rw-rw-r-- 1 jay jay 5321567 Feb 15 11:28 v1_vue_full_attack.csv
```

(Optional) Prepare Full Dataset

If you want to prepare your own full dataset run these commands with the REST API running locally:

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/vue/configs/vue-prepare-v1.
↪json
/opt/antinex/api/tests/build-new-dataset.py
```

Confirm Dataset is Ready

```
/opt/antinex/antinex-datasets/tools/describe-v1-training.py /opt/antinex/antinex-
↪datasets/v1/webapps/vue/training-ready/v1_vue_cleaned.csv
```

Hopefully your dataset has both attack and non-attack records like:

```
2018-02-15 11:29:11,207 - describe-training-data - INFO - total records=25600␣
↪attack=4400 nonattack=21200 percent_attack=17.19% percent_nonattack=82.81%
```

What you don't want to see is this in the output:

```
2018-02-15 08:47:41,389 - describe-training-data - INFO - total records=21200␣
↪attack=0 nonattack=21200 percent_attack=0.00% percent_nonattack=100.00%
```

That means the prepare step failed to add the attack data into the dataset correctly. Please go back to the `Prepare Dataset` step and review paths to the files are correct.

Train Dataset

```
source ~/.venvs/venvdrfpipeline/bin/activate
export TEST_DATA=/opt/antinex/antinex-datasets/v1/webapps/vue/configs/vue-train-v1.
↪json
/opt/antinex/api/tests/create-keras-dnn.py
```

From the logs taken during creation of this doc, the model is 83% accurate at predicting attack records.

INFO:create-keras-dnn:Logging in user url=http://localhost:8010/api-token-auth/ INFO:create-keras-dnn:logged in user=root token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MTg3M INFO:create-keras-dnn:building post data INFO:create-keras-dnn:Running ML Job url=http://localhost:8010/ml/ test_data={'csv_file': '/opt/antinex/antinex-datasets/v1/webapps/vue/training-ready/v1_vue_cleaned.csv', 'meta_file': '/opt/antinex/antinex-datasets/v1/webapps/vue/training-ready/cleaned_v1_vue_metadata.json', 'title': 'Vue - Keras DNN - Dataset v1', 'desc': 'Training Vue DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'predict_feature': 'label_value', 'training_data': '{}', 'pre_proc': '{}', 'post_proc': '{}', 'meta_data': '{}', 'version': 1} INFO:create-keras-dnn:SUCCESS - Post Response status=201 reason=Created INFO:create-keras-dnn:{'job': {'id': 18, 'user_id': 1,

'user_name': 'root', 'title': 'Vue - Keras DNN - Dataset v1', 'desc': 'Training Vue DNN using Attack and Non-attack data captured using the network-pipeline', 'ds_name': 'cleaned', 'algo_name': 'dnn', 'ml_type': 'keras', 'status': 'initial', 'control_state': 'active', 'predict_feature': 'label_value', 'training_data': {}, 'pre_proc': {}, 'post_proc': {}, 'meta_data': {}, 'tracking_id': 'ml_ad7d1a31-c7b3-47ec-9c69-3e55a12c7bf3', 'version': 1, 'created': '2018-02-15 19:29:25', 'updated': '2018-02-15 19:29:25', 'deleted': ''}, 'results': {'id': 14, 'user_id': 1, 'user_name': 'root', 'job_id': 18, 'status': 'finished', 'version': 1, 'acc_data': {**'accuracy': 83.1640625**}, 'error_data': None, 'model_json': '{"class_name": "Sequential", "config": [{"class_name": "Dense", "config": {"name": "dense_4", "trainable": true, "batch_input_shape": [null, 68], "dtype": "float32", "units": 8, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name": "dense_5", "trainable": true, "units": 6, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name": "dense_6", "trainable": true, "units": 1, "activation": "sigmoid", "use_bias": true, "kernel_initializer": {"class_name": "RandomUniform", "config": {"minval": -0.05, "maxval": 0.05, "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}], "keras_version": "2.1.4", "backend": "tensorflow"}', 'model_weights': {'weights': '[[[0.0964929983,0.0837596282,0.0513114333,-
0.0693184286,-0.0125864763,0.0574291162,-0.0161194559,0.0421481095],[0.0764627904,0.0430329069,-
0.0191652663,0.0074863322,-0.0002829469,0.1027121916,-0.0398681201,-0.0201492198],[-0.056526497,-
0.020370299,-0.0472459234,-0.0560302138,0.0236291252,-0.0822693929,-0.0890297815,-0.0480410382],[-
0.0064245733,0.02736664,-0.0115312617,-0.0152309267,0.0057650399,0.0353901014,-0.0304064881,-
0.014901977],[0.0062483978,0.0532563478,0.0526493713,0.00063548,0.0492007136,0.0401249528,-
0.0732960626,0.0336591713],[0.0808463618,0.0222757515,0.0604787916,-0.0440429412,0.0354416184,0.1061460897,0.077620305
0.0438821949],[-0.0034223702,-0.031368304,-0.0190553498,-0.023251174,0.0308189727,-
0.0795371532,-0.0777133033,0.01265141],[-0.0686491504,0.0194620099,-0.0424295217,-
0.0240028482,-0.0144291371,-0.0117279524,-0.0344395377,-0.0395662189],[-0.0660949051,-
0.0486420654,0.0097997449,0.0116964141,0.0569563545,-0.0330945961,-0.0009582887,0.0109461136],[-
0.0271332134,-0.0048795654,0.0104634706,-0.0352833308,0.0343606696,-0.0715944916,-
0.0167990662,0.0252644829],[-0.0699779168,-0.0437729359,-0.014376387,-0.0475340076,-
0.010354978,-0.0230973363,-0.0009265427,-0.0299541708],[-0.0219055898,-0.0522471406,-
0.0189603493,-0.0512510501,-0.0005330394,-0.0555729195,-0.0520991385,-0.0211805589],[-
0.0404028483,0.020369729,0.0023353414,0.0119193504,-0.0124673871,-0.0115194013,0.0054978556,-
0.028604554],[-0.0511647239,-0.0261075459,-0.0467265062,0.0189402122,0.0549614727,-0.0663635805,-
0.0279217064,0.0042245188],[-0.0806629583,0.0107657518,0.0117750419,-0.0536228344,0.0310879052,-
0.0637154654,-0.0681177229,0.0161142722],[-0.0744192228,-0.0230462849,0.0186079368,-
0.0033951802,0.0216188282,-0.062285006,-0.062960647,0.0300793424],[-0.0315857232,-0.0647951886,-
0.0148812886,-0.0678566247,0.023202572,-0.0827157721,-0.013793733,-0.0130123906],[-
0.0297478233,-0.0688403174,0.0396807306,-0.0757085979,0.031263493,-0.0002234936,-0.0909571871,-
0.0323583074],[-0.0240834001,-0.0398418121,-0.0064964825,-0.0367884561,0.0719116777,-
0.0823321193,-0.0645215511,0.0338410847],[-0.0384184793,-0.0576236956,-0.019627776,-0.0506528094,-
0.0163098071,-0.0409412533,-0.041446805,0.0172828659],[-0.005851123,-0.0429788493,-0.0252392255,-
0.073766917,0.0097925663,-0.0794945806,-0.0849412307,0.0014261433],[-0.0512101054,0.0005189396,-
0.0271312632,-0.0270303208,0.0178172924,-0.0880423188,-0.0889234915,-0.0421689749],[-
0.0717964247,-0.0206637289,-0.0067394814,-0.0284654628,0.0247239321,-0.0477216393,-
0.036414776,0.0200105682],[-0.0943818614,-0.0469460264,0.0279882532,-0.0619334579,0.0121557433,-
0.0768882185,0.0002953591,0.0396316908],[-0.0686878562,-0.0194259193,0.0096589588,-0.0417566299,-
0.0048376634,-0.0504943505,-0.0156595726,0.027880745],[-0.0523356088,-0.0725596026,0.0059417831,-
0.0251557883,-0.0038264154,-0.0179003626,-0.0591997802,0.0274024624],[-0.0341782831,-
0.0221797936,0.0042649712,-0.0477346219,0.0461222231,-0.0023178295,-0.03149179,0.0011711826],[-
0.073523894,0.020247506,0.0248407796,0.0047891312,0.0758228377,-0.0405860767,-
0.0173077062,0.0444364958],[0.0393536985,0.0223959032,0.0194161702,0.0163059514,0.0072390498,0.0256230235,0.0177581478

0.0273305401],[-0.0929955989,0.0247371849,-0.0009080072,-0.0237545576,0.0517009124,-
0.019001672,-0.0027636925,0.0225250944],[-0.0503246076,0.0514643453,-0.034101136,0.071626991,-
0.0102834394,-0.0852171034,0.0230983458,-0.004897465],[-0.0262203328,-0.0664628223,-0.0034848654,-
0.0033428837,0.0001543516,-0.0749761909,-0.024853928,0.0465056859],[-0.052637279,-0.0315136686,-
0.0437926725,-0.0341987237,0.0392551571,-0.0518464595,-0.0266457852,0.0146500943],[-
0.0341906585,-0.0460761786,-0.0180298146,-0.0055014049,0.0707753003,-0.0777451321,-
0.0366662145,0.0145203276],[-0.0122610182,0.014549084,-0.0121618919,0.0285700168,0.0231714714,-
0.0371087231,0.0220328253,-0.0466214307],[-0.0198020525,0.0109219337,0.0416364111,-
0.0783106089,0.0728293508,-0.0356889144,-0.0880245343,0.0330911092],[-0.0854380801,-
0.0669428557,-0.0214084946,-0.0081647681,0.0233404748,-0.0217991099,-0.0376962982,-
0.042799335],[-0.0452309586,-0.0024355939,0.0175608397,-0.0052607814,0.0183441378,-0.0620779246,-
0.0699491203,-0.0118549634],[-0.0959726647,0.0476075932,-0.0379351601,0.008159698,0.0384081416,-
0.0612908602,-0.0189263411,-0.0454253554],[-0.0053411066,-0.0094355131,-0.0510591529,-
0.0221638251,0.0046492857,-0.0514930561,-0.0705555007,-0.020327853],[0.0661738142,-
0.0029260954,-0.0595169291,0.0101059973,-0.0022012119,0.049445834,0.0558575578,-0.012022947],[-
0.0830700099,0.0212585274,-0.0389494337,-0.0166049339,0.0477221608,-0.0867467299,-
0.0052994671,0.0331948586],[-0.0076975222,0.0475708768,-0.022361204,0.0159589462,0.0034970916,-
0.0780191272,0.0727450103,0.0052784681],[-0.0611799173,-0.0223675836,-0.0501373671,0.0420941673,-
0.0056689265,-0.0148042021,0.0729248822,0.0337781236],[-0.0021687131,0.0187189635,-
0.0138026681,0.0335950628,0.0171724465,-0.0114344545,0.0064723557,0.014866543],[-
0.0029786164,0.0592048354,-0.0431581549,0.0631211698,-0.0203074273,-0.0943428278,0.0607772991,0.0175965521],[-
0.0953431055,0.0004848846,-0.055826772,0.0066280495,0.0397272743,-0.056723319,-
0.0556911081,0.0250450671],[-0.0736271068,-0.0255948883,0.0372888483,0.0277335718,-
0.0149522563,-0.0228647534,-0.0830083936,0.0032964509],[-0.062650837,-0.0202989262,-
0.0077688964,-0.0318020023,0.0132382195,0.0001132841,-0.043601539,0.0263965204],[-0.0097646713,-
0.0558951087,-0.039418485,-0.0255730916,0.013469019,-0.0136131318,-0.0810244158,0.0127038872],[-
0.0612644814,-0.0250901058,0.0282239243,0.0373365022,0.0209282301,-0.0226692595,-
0.0864989087,-0.0179349761],[-0.0924655274,0.0218485277,-0.0298216268,-0.0057617133,-
0.0296175014,-0.0120711923,-0.0545658804,0.0150848003],[-0.0751573443,-0.0002987361,-
0.0057580415,0.0093203867,-0.0140915299,-0.0325664468,-0.0779319704,0.0292233229],[-
0.0079788342,0.0444301628,0.0362950899,0.0386882909,0.0018038042,-0.048700463,0.0057461374,-
0.041268073],[-0.0221946761,-0.050303746,0.0298334453,-0.0346691869,0.0488329567,-
0.0676441416,-0.0486205034,0.0179271474],[-0.0806570575,-0.0448458456,-0.0144681996,-
0.0426183194,-0.0061101187,-0.0247607604,0.0180130191,0.0609768555],[-0.0479587466,-
0.0541830994,0.0020282909,0.0168216806,0.0308838021,-0.0877256617,-0.013448243,0.0173823014],[-
0.065826796,-0.0472236201,-0.0440701954,-0.002680534,0.0611480027,-0.0693870261,-0.0684818774,-
0.0339303389],[-0.0024183108,-0.0320040435,0.0125231585,-0.0658573955,0.0589915328,-
0.0091869226,-0.0402693488,-0.001413801],[-0.0122722089,-0.0643372461,-0.0261148885,-
0.0108729294,0.0712984875,-0.0828262791,-0.0310833678,0.0446092822],[-0.0557892919,0.0059863473,-
0.0507959016,-0.0419340506,0.0633320585,-0.0045545883,-0.0016710822,0.0357435755],[-0.0203432944,-
0.0151491342,-0.0224984549,-0.0745795816,0.0090752272,-0.0742223784,0.0066466844,-0.0276446585],[-
0.0677567497,0.0357993916,0.0044564921,-0.0539436862,-0.0110100135,0.0332751572,0.0323642008,0.0356529839],[-
0.0750699639,-0.0045500868,-0.0511593483,-0.0463683642,0.0122727454,-0.0276447199,0.0391405001,0.0306293499],[-
0.0934526324,-0.039043989,-0.0426228829,-0.0394503661,-0.0389413275,-0.0964915231,0.0093822209,0.0348296985],[-
0.0690298229,-0.0508973859,0.0259087458,-0.0228281599,-0.0076745511,-0.0086752577,-
0.0901542827,0.0304441247],[-0.0338399112,-0.0072802799,-0.0317518972,-0.0181066915,0.0328182429,-
0.0979935005,0.0129396953,0.0413899347],[-0.0805611685,0.0028219493,0.0320406146,0.0552994162,-
0.0416063294,-0.0664851591,0.0082092546,-0.0352456868]],[0.0499761477,0.0235879347,0.0078678448,0.0307559464,-
0.0262398124,0.0492662415,0.0417507663,0.0019115364],[[-0.0497738235,0.082566157,0.0858044401,0.0807759464,-
0.0281407479,-0.0240644943],[-0.0080805197,0.0533784255,0.0423657559,0.0844753161,-
0.0132133355,0.0310063623],[0.0433566235,0.024253225,0.0388046354,0.1547681987,-0.0076785884,-
0.0361264572],[-0.0191759467,0.0495774075,0.096243605,0.1220689639,0.0066543957,-
0.0348133594],[-0.0043437853,0.000867462,0.0509670079,0.0299713202,-0.0031723627,-
0.005397758],[-0.0177731514,0.0803218558,0.0660849884,0.13484326,-0.0441330299,0.0192133877],[-

0.0007352605,0.0217755958,0.0690242201,0.0731370077,0.0624744706,-0.0324748941],[-
0.0474479459,0.0553567745,-0.0198244397,-0.010604023,0.0421101451,0.0116976937]],[0.0,0.0703127161,0.0674306825,0.05849
0.0126846386],[[0.0055085532],[-0.1386123598],[-0.0565749854],[-0.0520785004],[-
0.066370979],[0.0197858457]],[-0.0406405143]]'}, 'acc_image_file': '/media/sf_shared/accuracy_job_18_result_14.png',
'created': '2018-02-15 19:30:36', 'updated': '2018-02-15 19:30:36', 'deleted': ''}}

Get the Deep Neural Network Accuracy, JSON and Weights

This will display all the recent training runs in a list sorted by newest.

```
/opt/antinex/api/tests/get-recent-results.py
```

Here's the training node in the list from the run above (yours will look a little different):

```
{
    "acc_data": {
        "accuracy": 83.1640625
    },
    "acc_image_file": "/media/sf_shared/accuracy_job_18_result_14.png",
    "created": "2018-02-15 19:30:36",
    "deleted": "",
    "error_data": null,
    "id": 14,
    "job_id": 18,
    "model_json": "{\"class_name\": \"Sequential\", \"config\": [{\"class_name\": \
↪"Dense\", \"config\": {\"name\": \"dense_4\", \"trainable\": true, \"batch_input_
↪shape\": [null, 68], \"dtype\": \"float32\", \"units\": 8, \"activation\": \"relu\",
↪ \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \"RandomUniform\", \
↪"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}, \"bias_
↪initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_regularizer\":␣
↪null, \"bias_regularizer\": null, \"activity_regularizer\": null, \"kernel_
↪constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\", \
↪"config\": {\"name\": \"dense_5\", \"trainable\": true, \"units\": 6, \"activation\
↪": \"relu\", \"use_bias\": true, \"kernel_initializer\": {\"class_name\": \
↪"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\": null}}
↪, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \"kernel_
↪regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\": null, \
↪"kernel_constraint\": null, \"bias_constraint\": null}}, {\"class_name\": \"Dense\",
↪ \"config\": {\"name\": \"dense_6\", \"trainable\": true, \"units\": 1, \
↪"activation\": \"sigmoid\", \"use_bias\": true, \"kernel_initializer\": {\"class_
↪name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\": 0.05, \"seed\
↪": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\": {}}, \
↪"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_regularizer\":␣
↪null, \"kernel_constraint\": null, \"bias_constraint\": null}}], \"keras_version\":␣
↪\"2.1.4\", \"backend\": \"tensorflow\"}",
    "model_weights": {
        "weights": "[[[0.0964929983,0.0837596282,0.0513114333,-0.0693184286,-0.
↪0125864763,0.0574291162,-0.0161194559,0.0421481095],[0.0764627904,0.0430329069,-0.
↪0191652663,0.0074863322,-0.0002829469,0.1027121916,-0.0398681201,-0.0201492198],[-0.
↪056526497,-0.020370299,-0.0472459234,-0.0560302138,0.0236291252,-0.0822693929,-0.
↪0890297815,-0.0480410382],[-0.0064245733,0.02736664,-0.0115312617,-0.0152309267,0.
↪0057650399,0.0353901014,-0.0304064881,-0.014901977],[0.0062483978,0.0532563478,0.
↪0526493713,0.00063548,0.0492007136,0.0401249528,-0.0732960626,0.0336591713],[0.
↪0808463618,0.0222757515,0.0604787916,-0.0440429412,0.0354416184,0.1061460897,0.
↪0776203051,-0.0438821949],[-0.0034223702,-0.031368304,-0.0190553498,-0.023251174,0.
↪0308189727,-0.0795371532,-0.0777133033,0.01265141],[-0.0686491504,0.0194620099,-0.
↪0424295217,-0.0240028482,-0.0144291371,-0.0117279524,-0.0344395377,-0.0395662189],[-
↪0.0660949051,-0.0486420654,0.0097997449,0.0116964141,0.0569563545,-0.0[illegible],0.
↪0009582887,0.0109461136],[-0.0271332134,-0.0048795654,0.0104634706,-0.0352833308,0.
↪0343606696,-0.0715944916,-0.0167990662,0.0252644829],[-0.0699779168,-0.0437729359,-
↪0.0[illegible],-0.0[illegible]5340076,-0.010354978,-0.0230973363,-0.0009265427,-0.0299541708],[-
↪0.0219055898,-0.0522471406,-0.0189603493,-0.0512510501,-0.0005330394,-0.0555729195,-
↪0.0520991385,-0.0211805589],[-0.0404028483,0.020369729,0.0023353414,0.0119193504,-0.
↪0124673871,-0.0115194013,0.0054978556,-0.028604554],[-0.0511647239,-0.0261075459,-0.
```

(continues on next page)

(continued from previous page)

```
    },
    "status": "finished",
    "updated": "2018-02-15 19:30:36",
    "user_id": 1,
    "user_name": "root",
    "version": 1
}
```

Here is the full HTTP request for preparing a dataset. For reference:

4.4.6 Inputs

ds_glob_path is where the CSV files are on disk.

4.4.7 Outputs

clean_file will be the training-ready CSV file create **full_file** is a full CSV file from the prepared datasets, this means it will have things like dates, strings and other non-numeric values that make it not training-ready without additional clean up steps. Please use the CSV file created in the **clean_file** value for training a Deep Neural Network.

will be an output the from the `packet-redis.py` script running in the **pipeline** container. The script writes csv files

```
{
    "title": "Prepare new Dataset from recordings",
    "desc": "",
    "ds_name": "new_recording",
    "ds_glob_path": "/opt/antinex/datasets/*/*.csv",
    "ds_dir": "/opt/antinex/datasets",
    "full_file": "/tmp/fulldata_attack_scans.csv",
    "clean_file": "/tmp/cleaned_attack_scans.csv",
    "meta_suffix": "metadata.json",
    "output_dir": "/tmp/",
    "pipeline_files": {
        "attack_files": []
    },
    "meta_data": {},
    "post_proc": {
        "drop_columns": [
            "src_file",
            "raw_id",
            "raw_load",
            "raw_hex_load",
            "raw_hex_field_load",
            "pad_load",
            "eth_dst",
            "eth_src",
            "ip_dst",
            "ip_src"
        ],
        "predict_feature": "label_name"
    },
    "label_rules": {
        "set_if_above": 85,
```

(continues on next page)

(continued from previous page)

```
        "labels": [
            "not_attack",
            "attack"
        ],
        "label_values": [
            0,
            1
        ]
    },
    "version": 1
}
```

4.4.8 Prepare a Dataset using Curl

```
auth_header="Authorization: JWT ${token}"
curl -s -X POST \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header "${auth_header}" \
    -d '{ "title": "Prepare new Dataset from recordings", "desc": "", "ds_name": "new_
↪recording", "ds_glob_path": "/opt/antinex/datasets/*/*.csv", "ds_dir": "/opt/
↪antinex/datasets", "full_file": "/tmp/fulldata_attack_scans.csv", "clean_file": "/
↪tmp/cleaned_attack_scans.csv", "meta_suffix": "metadata.json", "output_dir": "/tmp/
↪", "pipeline_files": { "attack_files": [] }, "meta_data": {}, "post_proc": { "drop_
↪columns": [ "src_file", "raw_id", "raw_load", "raw_hex_load", "raw_hex_field_load",
↪"pad_load", "eth_dst", "eth_src", "ip_dst", "ip_src" ], "predict_feature": "label_
↪name" }, "label_rules": { "set_if_above": 85, "labels": [ "not_attack", "attack" ],
↪"label_values": [ 0, 1 ] }, "version": 1 }' \
    'http://0.0.0.0:8010/mlprepare/'

{"id":1,"user_id":1,"user_name":"root","status":"initial","control_state":"active",
↪"title":"Prepare new Dataset from recordings","desc":"no desc","full_file":"/tmp/
↪fulldata_attack_scans.csv","clean_file":"/tmp/cleaned_attack_scans.csv","meta_suffix
↪":"metadata.json","output_dir":"/tmp/","ds_dir":"/opt/antinex/datasets","ds_glob_
↪path":"/opt/antinex/datasets/*/*.csv","pipeline_files":{"attack_files":[]},"post_
↪proc":{"drop_columns":["src_file","raw_id","raw_load","raw_hex_load","raw_hex_field_
↪load","pad_load","eth_dst","eth_src","ip_dst","ip_src"],"predict_feature":"label_
↪name"},"label_rules":{"set_if_above":85,"labels":["not_attack","attack"],"label_
↪values":[0,1]},"tracking_id":"prep_fcd155e3-bd99-46a5-86d9-957fc7a95a8a","version
↪":1,"created":"2018-03-30 16:06:49","updated":"2018-03-30 16:06:49","deleted":""}
```

4.4.9 Check the Newly Prepared Dataset Files Exist

```
ls -l /tmp/*.csv
-rw-r--r-- 1 jay jay   145580 Mar 30 09:07 /tmp/cleaned_attack_scans.csv
-rw-r--r-- 1 jay jay 26478291 Mar 30 09:07 /tmp/fulldata_attack_scans.csv
```

There is also metadata and dataset debugging information in the created JSON files:

```
ls -l /tmp/*.json
-rw-r--r-- 1 jay jay 1498 Mar 30 09:07 /tmp/cleaned_metadata.json
-rw-r--r-- 1 jay jay 2669 Mar 30 09:07 /tmp/fulldata_metadata.json
```

4.4.10 Get Prepared Dataset Record from the Database using Curl

```
auth_header="Authorization: JWT ${token}"
curl -s -X GET \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header "${auth_header}" \
    'http://0.0.0.0:8010/mlprepare/1/'

{"id":1,"user_id":1,"user_name":"root","status":"finished","control_state":"finished",
↪"title":"Prepare new Dataset from recordings","desc":"no desc","full_file":"/tmp/
↪fulldata_attack_scans.csv","clean_file":"/tmp/cleaned_attack_scans.csv","meta_suffix
↪":"metadata.json","output_dir":"/tmp/","ds_dir":"/opt/antinex/datasets","ds_glob_
↪path":"/opt/antinex/datasets/*/*.csv","pipeline_files":["/opt/antinex/datasets/
↪react-redux/netdata-2018-01-29-13-36-35.csv","/opt/antinex/datasets/spring/netdata-
↪2018-01-29-15-00-12.csv","/opt/antinex/datasets/vue/netdata-2018-01-29-14-12-44.csv
↪","/opt/antinex/datasets/django/netdata-2018-01-28-23-12-13.csv","/opt/antinex/
↪datasets/django/netdata-2018-01-28-23-06-05.csv","/opt/antinex/datasets/flask-
↪restplus/netdata-2018-01-29-11-30-02.csv"],"post_proc":{"drop_columns":["src_file",
↪"raw_id","raw_load","raw_hex_load","raw_hex_field_load","pad_load","eth_dst","eth_
↪src","ip_dst","ip_src"],"ignore_features":["label_name","src_file","raw_id","raw_
↪load","raw_hex_field_load","pad_load","eth_dst","eth_src","ip_dst","ip_src"],
↪"predict_feature":"label_name","feature_to_predict":"label_name","features_to_
↪process":["arp_id","dns_id","eth_id","eth_type","icmp_id","idx","ip_id","ip_ihl",
↪"ip_len","ip_tos","ip_version","ipvsix_id","label_value","pad_id","tcp_dport","tcp_
↪fields_options.MSS","tcp_fields_options.NOP","tcp_fields_options.SAckOK","tcp_
↪fields_options.Timestamp","tcp_fields_options.WScale","tcp_id","tcp_seq","tcp_sport
↪","udp_id","label_name"]},"label_rules":{"labels":["not_attack","attack"],"label_
↪values":[0,1],"set_if_above":85},"tracking_id":"prep_fcd155e3-bd99-46a5-86d9-
↪957fc7a95a8a","version":1,"created":"2018-03-30 16:06:49","updated":"2018-03-30␣
↪16:07:07","deleted":""}
```

4.5 Train a Deep Neural Network with a Dataset

If you want to use the AntiNex Datasets repository you will need to clone the repository locally.

```
git clone https://github.com/jay-johnson/antinex-datasets /opt/antinex/antinex-
↪datasets
```

Here is the full HTTP request for training a new Deep Neural Network from a dataset on disk:

```
{
    "label": "Full-Django-AntiNex-Simple-Scaler-DNN",
    "dataset": "/opt/antinex/antinex-datasets/v1/webapps/django/training-ready/v1_
↪django_cleaned.csv",
    "ml_type": "classification",
    "publish_to_core": true,
    "predict_feature": "label_value",
    "features_to_process": [
        "idx",
        "arp_hwlen",
        "arp_hwtype",
        "arp_id",
        "arp_op",
        "arp_plen",
```

(continues on next page)

(continued from previous page)

```
        "arp_ptype",
        "dns_default_aa",
        "dns_default_ad",
        "dns_default_an",
        "dns_default_ancount",
        "dns_default_ar",
        "dns_default_arcount",
        "dns_default_cd",
        "dns_default_id",
        "dns_default_length",
        "dns_default_ns",
        "dns_default_nscount",
        "dns_default_opcode",
        "dns_default_qd",
        "dns_default_qdcount",
        "dns_default_qr",
        "dns_default_ra",
        "dns_default_rcode",
        "dns_default_rd",
        "dns_default_tc",
        "dns_default_z",
        "dns_id",
        "eth_id",
        "eth_type",
        "icmp_addr_mask",
        "icmp_code",
        "icmp_gw",
        "icmp_id",
        "icmp_ptr",
        "icmp_seq",
        "icmp_ts_ori",
        "icmp_ts_rx",
        "icmp_ts_tx",
        "icmp_type",
        "icmp_unused",
        "ip_id",
        "ip_ihl",
        "ip_len",
        "ip_tos",
        "ip_version",
        "ipv6_fl",
        "ipv6_hlim",
        "ipv6_nh",
        "ipv6_plen",
        "ipv6_tc",
        "ipv6_version",
        "ipvsix_id",
        "pad_id",
        "tcp_dport",
        "tcp_fields_options.MSS",
        "tcp_fields_options.NOP",
        "tcp_fields_options.SAckOK",
        "tcp_fields_options.Timestamp",
        "tcp_fields_options.WScale",
        "tcp_id",
        "tcp_seq",
        "tcp_sport",
```

(continues on next page)

(continued from previous page)

```
        "udp_dport",
        "udp_id",
        "udp_len",
        "udp_sport"
    ],
    "ignore_features": [
    ],
    "sort_values": [
    ],
    "seed": 42,
    "test_size": 0.2,
    "batch_size": 32,
    "epochs": 15,
    "num_splits": 2,
    "loss": "binary_crossentropy",
    "optimizer": "adam",
    "metrics": [
        "accuracy"
    ],
    "histories": [
        "val_loss",
        "val_acc",
        "loss",
        "acc"
    ],
    "model_desc": {
        "layers": [
            {
                "num_neurons": 200,
                "init": "uniform",
                "activation": "relu"
            },
            {
                "num_neurons": 1,
                "init": "uniform",
                "activation": "sigmoid"
            }
        ]
    },
    "label_rules": {
        "labels": [
            "not_attack",
            "not_attack",
            "attack"
        ],
        "label_values": [
            -1,
            0,
            1
        ]
    },
    "version": 1
}
```

4.5.1 Train a Deep Neural Network with Curl

This example created Deep Neural Network by Job ID **3** with Results ID **3**.

```
auth_header="Authorization: JWT ${token}"
curl -s -X POST \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header "${auth_header}" \
    -d '{ "label": "Full-Django-AntiNex-Simple-Scaler-DNN", "dataset": "/opt/antinex/
↪antinex-datasets/v1/webapps/django/training-ready/v1_django_cleaned.csv", "ml_type
↪": "classification", "publish_to_core": true, "predict_feature": "label_value",
↪"features_to_process": [ "idx", "arp_hwlen", "arp_hwtype", "arp_id", "arp_op", "arp_
↪plen", "arp_ptype", "dns_default_aa", "dns_default_ad", "dns_default_an", "dns_
↪default_ancount", "dns_default_ar", "dns_default_arcount", "dns_default_cd", "dns_
↪default_id", "dns_default_length", "dns_default_ns", "dns_default_nscount", "dns_
↪default_opcode", "dns_default_qd", "dns_default_qdcount", "dns_default_qr", "dns_
↪default_ra", "dns_default_rcode", "dns_default_rd", "dns_default_tc", "dns_default_z
↪", "dns_id", "eth_id", "eth_type", "icmp_addr_mask", "icmp_code", "icmp_gw", "icmp_
↪id", "icmp_ptr", "icmp_seq", "icmp_ts_ori", "icmp_ts_rx", "icmp_ts_tx", "icmp_type",
↪ "icmp_unused", "ip_id", "ip_ihl", "ip_len", "ip_tos", "ip_version", "ipv6_fl",
↪"ipv6_hlim", "ipv6_nh", "ipv6_plen", "ipv6_tc", "ipv6_version", "ipvsix_id", "pad_id
↪", "tcp_dport", "tcp_fields_options.MSS", "tcp_fields_options.NOP", "tcp_fields_
↪options.SAckOK", "tcp_fields_options.Timestamp", "tcp_fields_options.WScale", "tcp_
↪id", "tcp_seq", "tcp_sport", "udp_dport", "udp_id", "udp_len", "udp_sport" ],
↪"ignore_features": [ ], "sort_values": [ ], "seed": 42, "test_size": 0.2, "batch_
↪size": 32, "epochs": 15, "num_splits": 2, "loss": "binary_crossentropy", "optimizer
↪": "adam", "metrics": [ "accuracy" ], "histories": [ "val_loss", "val_acc", "loss",
↪"acc" ], "model_desc": { "layers": [ { "num_neurons": 200, "init": "uniform",
↪"activation": "relu" }, { "num_neurons": 1, "init": "uniform", "activation":
↪"sigmoid" } ] }, "label_rules": { "labels": [ "not_attack", "not_attack", "attack"␣
↪], "label_values": [ -1, 0, 1 ] }, "version": 1 }' \
    'http://0.0.0.0:8010/ml/'

{"job":{"id":3,"user_id":1,"user_name":"root","title":"Full-Django-AntiNex-Simple-
↪Scaler-DNN","desc":null,"ds_name":"Full-Django-AntiNex-Simple-Scaler-DNN","algo_name
↪":"Full-Django-AntiNex-Simple-Scaler-DNN","ml_type":"classification","status":
↪"initial","control_state":"active","predict_feature":"label_value","predict_manifest
↪":{"job_id":3,"result_id":3,"ml_type":"classification","test_size":0.2,"epochs":15,
↪"batch_size":32,"num_splits":2,"loss":"binary_crossentropy","metrics":["accuracy"],
↪"optimizer":"adam","histories":["val_loss","val_acc","loss","acc"],"seed":42,
↪"training_data":{},"csv_file":null,"meta_file":null,"use_model_name":"Full-Django-
↪AntiNex-Simple-Scaler-DNN","dataset":"/opt/antinex/antinex-datasets/v1/webapps/
↪django/training-ready/v1_django_cleaned.csv","predict_rows":null,"apply_scaler
↪":true,"predict_feature":"label_value","features_to_process":["idx","arp_hwlen",
↪"arp_hwtype","arp_id","arp_op","arp_plen","arp_ptype","dns_default_aa","dns_default_
↪ad","dns_default_an","dns_default_ancount","dns_default_ar","dns_default_arcount",
↪"dns_default_cd","dns_default_id","dns_default_length","dns_default_ns","dns_
↪default_nscount","dns_default_opcode","dns_default_qd","dns_default_qdcount","dns_
↪default_qr","dns_default_ra","dns_default_rcode","dns_default_rd","dns_default_tc",
↪"dns_default_z","dns_id","eth_id","eth_type","icmp_addr_mask","icmp_code","icmp_gw",
↪"icmp_id","icmp_ptr","icmp_seq","icmp_ts_ori","icmp_ts_rx","icmp_ts_tx","icmp_type",
↪"icmp_unused","ip_id","ip_ihl","ip_len","ip_tos","ip_version","ipv6_fl","ipv6_hlim",
↪"ipv6_nh","ipv6_plen","ipv6_tc","ipv6_version","ipvsix_id","pad_id","tcp_dport",
↪"tcp_fields_options.MSS","tcp_fields_options.NOP","tcp_fields_options.SAckOK","tcp_
↪fields_options.Timestamp","tcp_fields_options.WScale","tcp_id","tcp_seq","tcp_sport
↪","udp_dport","udp_id","udp_len","udp_sport"],"ignore_features":[],"sort_values":[],
↪"model_desc":{"layers":[{"num_neurons":200,"init":"uniform","activation":"relu"},{
↪"num_neurons":1,"init":"uniform","activation":"sigmoid"}]},"label_rules":{"labels":[
↪"not_attack","not_attack","attack"],"label_values":[-1,0,1]},"post_proc_rules":null,
↪"model_weights_file":"/tmp/ml_weights_job_3_result_3.h5","verbose":1,"version":1,
↪"publish_to_core":true,"worker_result_node":{"source":"drf","auth_url":"redis://
↪localhost:6379/9","ssl_options":{},"exchange":"drf_network_pipeline.pipeline.tasks.
↪task_ml_process_results","exchange_type":"topic","routing_key":"drf_network_
↪pipeline.pipeline.tasks.task_ml_process_results","queue":"drf_network_pipeline.
↪pipeline.tasks.task_ml_process_results","delivery_mode":2,"task_name":"drf_network
```

(continues on next page)

(continued from previous page)

4.5.2 Get a Deep Neural Network Job Record with Curl

Get the example Deep Neural Network Job by ID **3**.

```
auth_header="Authorization: JWT ${token}"
curl -s -X GET \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header "${auth_header}" \
    'http://0.0.0.0:8010/ml/3/'
```

```
{"id":3,"user_id":1,"user_name":"root","title":"Full-Django-AntiNex-Simple-Scaler-DNN
↪","desc":null,"ds_name":"Full-Django-AntiNex-Simple-Scaler-DNN","algo_name":"Full-
↪Django-AntiNex-Simple-Scaler-DNN","ml_type":"classification","status":"finished",
↪"control_state":"finished","predict_feature":"label_value","predict_manifest":{"loss
↪":"binary_crossentropy","seed":42,"epochs":15,"job_id":3,"dataset":"/opt/antinex/
↪antinex-datasets/v1/webapps/django/training-ready/v1_django_cleaned.csv","metrics":[
↪"accuracy"],"ml_type":"classification","verbose":1,"version":1,"csv_file":null,
↪"histories":["val_loss","val_acc","loss","acc"],"meta_file":null,"optimizer":"adam",
↪"result_id":3,"test_size":0.2,"batch_size":32,"model_desc":{"layers":[{"init":
↪"uniform","activation":"relu","num_neurons":200},{"init":"uniform","activation":
↪"sigmoid","num_neurons":1}]},"num_splits":2,"label_rules":{"labels":["not_attack",
↪"not_attack","attack"],"label_values":[-1,0,1]},"sort_values":[],"apply_scaler
↪":true,"predict_rows":null,"training_data":{},"use_model_name":"Full-Django-AntiNex-
↪Simple-Scaler-DNN","ignore_features":[],"post_proc_rules":null,"predict_feature":
↪"label_value","publish_to_core":true,"model_weights_file":"/tmp/ml_weights_job_3_
↪result_3.h5","worker_result_node":{"queue":"drf_network_pipeline.pipeline.tasks.
↪task_ml_process_results","source":"drf","auth_url":"redis://localhost:6379/9",
↪"exchange":"drf_network_pipeline.pipeline.tasks.task_ml_process_results","manifest":
↪{"job_id":3,"job_type":"train-and-predict","result_id":3},"task_name":"drf_network_
↪pipeline.pipeline.tasks.task_ml_process_results","routing_key":"drf_network_
↪pipeline.pipeline.tasks.task_ml_process_results","ssl_options":{},"delivery_mode":2,
↪"exchange_type":"topic"},"features_to_process":["idx","arp_hwlen","arp_hwtype","arp_
↪id","arp_op","arp_plen","arp_ptype","dns_default_aa","dns_default_ad","dns_default_
↪an","dns_default_ancount","dns_default_ar","dns_default_arcount","dns_default_cd",
↪"dns_default_id","dns_default_length","dns_default_ns","dns_default_nscount","dns_
↪default_opcode","dns_default_qd","dns_default_qdcount","dns_default_qr","dns_
↪default_ra","dns_default_rcode","dns_default_rd","dns_default_tc","dns_default_z",
↪"dns_id","eth_id","eth_type","icmp_addr_mask","icmp_code","icmp_gw","icmp_id","icmp_
↪ptr","icmp_seq","icmp_ts_ori","icmp_ts_rx","icmp_ts_tx","icmp_type","icmp_unused",
↪"ip_id","ip_ihl","ip_len","ip_tos","ip_version","ipv6_fl","ipv6_hlim","ipv6_nh",
↪"ipv6_plen","ipv6_tc","ipv6_version","ipvsix_id","pad_id","tcp_dport","tcp_fields_
↪options.MSS","tcp_fields_options.NOP","tcp_fields_options.SAckOK","tcp_fields_
↪options.Timestamp","tcp_fields_options.WScale","tcp_id","tcp_seq","tcp_sport","udp_
↪dport","udp_id","udp_len","udp_sport"]},"training_data":{},"pre_proc":{},"post_proc
↪":{},"meta_data":{},"tracking_id":"ml_4529bda5-2003-45ce-b08b-bfc48d6a008b","version
↪":1,"created":"2018-03-30 16:25:49","updated":"2018-03-30 16:26:49","deleted":""}
```

4.5.3 Get a Deep Neural Network Results with Curl

Get the example Deep Neural Network Training, Accuracy and Prediction Results by ID **3**.

Note: This will return all **30200** records so it can take a second

```
auth_header="Authorization: JWT ${token}"
curl -s -X GET \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header "${auth_header}" \
    'http://0.0.0.0:8010/mlresults/3/'

...

{"id":3,"user_id":1,"user_name":"root","job_id":3,"status":"finished","test_size":0.2,
↪"csv_file":null,"meta_file":null,"version":1,"acc_data":{"accuracy":99.
↪82615894039735},"error_data":null,"model_json":"{\"class_name\": \"Sequential\", \
↪"config\": [{\"class_name\": \"Dense\", \"config\": {\"name\": \"dense_1\", \
↪"trainable\": true, \"batch_input_shape\": [null, 67], \"dtype\": \"float32\", \
↪"units\": 200, \"activation\": \"relu\", \"use_bias\": true, \"kernel_initializer\
↪": {\"class_name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\":␣
↪0.05, \"seed\": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\
↪": {}}, \"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_
↪regularizer\": null, \"kernel_constraint\": null, \"bias_constraint\": null}}, {\
↪"class_name\": \"Dense\", \"config\": {\"name\": \"dense_2\", \"trainable\": true, \
↪"units\": 1, \"activation\": \"sigmoid\", \"use_bias\": true, \"kernel_initializer\
↪": {\"class_name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\":␣
↪0.05, \"seed\": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\
↪": {}}, \"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_
↪regularizer\": null, \"kernel_constraint\": null, \"bias_constraint\": null}}], \
↪"keras_version\": \"2.1.5\", \"backend\": \"tensorflow\"}","model_weights":{},"acc_
↪image_file":null,"predictions_json":{"predictions":[

... lots of prediction dictionaries
```

4.5.4 Make New Predictions with a Pre-trained Deep Neural Network with Curl

This example uses the pre-trained Deep Neural Network to make new predictions with a Job ID **4** with Results ID **4**.

```
auth_header="Authorization: JWT ${token}"
curl -s -X POST \
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header "${auth_header}" \
    -d '{ "label": "Full-Django-AntiNex-Simple-Scaler-DNN", "dataset": "/opt/antinex/
↪antinex-datasets/v1/webapps/django/training-ready/v1_django_cleaned.csv", "ml_type
↪": "classification", "publish_to_core": true, "predict_feature": "label_value",
↪"features_to_process": [ "idx", "arp_hwlen", "arp_hwtype", "arp_id", "arp_op", "arp_
↪plen", "arp_ptype", "dns_default_aa", "dns_default_ad", "dns_default_an", "dns_
↪default_ancount", "dns_default_ar", "dns_default_arcount", "dns_default_cd", "dns_
↪default_id", "dns_default_length", "dns_default_ns", "dns_default_nscount", "dns_
↪default_opcode", "dns_default_qd", "dns_default_qdcount", "dns_default_qr", "dns_
↪default_ra", "dns_default_rcode", "dns_default_rd", "dns_default_tc", "dns_default_z
↪", "dns_id", "eth_id", "eth_type", "icmp_addr_mask", "icmp_code", "icmp_gw", "icmp_
↪id", "icmp_ptr", "icmp_seq", "icmp_ts_ori", "icmp_ts_rx", "icmp_ts_tx", "icmp_type",
↪ "icmp_unused", "ip_id", "ip_ihl", "ip_len", "ip_tos", "ip_version", "ipv6_fl",
↪"ipv6_hlim", "ipv6_nh", "ipv6_plen", "ipv6_tc", "ipv6_version", "ipvsix_id", "pad_id
↪", "tcp_dport", "tcp_fields_options.MSS", "tcp_fields_options.NOP", "tcp_fields_
↪options.SAckOK", "tcp_fields_options.Timestamp", "tcp_fields_options.WScale", "tcp_
↪id", "tcp_seq", "tcp_sport", "udp_dport", "udp_id", "udp_len", "udp_sport" ],
↪"ignore_features": [ ], "sort_values": [ ], "seed": 42, "test_size": 0.2, "batch_
↪size": 32, "epochs": 15, "num_splits": 2, "loss": "binary_crossentropy", "optimizer
↪": "adam", "metrics": [ "accuracy" ], "histories": [ "val_loss", "val_acc", "loss",
↪"acc" ], "model_desc": { "layers": [ { "num_neurons": 200, "init": "uniform",
↪"activation": "relu" }, { "num_neurons": 1, "init": "uniform", "activation":
```

(continues on next page)

(continued from previous page)

```
    'http://0.0.0.0:8010/ml/'

{"job":{"id":4,"user_id":1,"user_name":"root","title":"Full-Django-AntiNex-Simple-
↪Scaler-DNN","desc":null,"ds_name":"Full-Django-AntiNex-Simple-Scaler-DNN","algo_name
↪":"Full-Django-AntiNex-Simple-Scaler-DNN","ml_type":"classification","status":
↪"initial","control_state":"active","predict_feature":"label_value","predict_manifest
↪":{"job_id":4,"result_id":4,"ml_type":"classification","test_size":0.2,"epochs":15,
↪"batch_size":32,"num_splits":2,"loss":"binary_crossentropy","metrics":["accuracy"],
↪"optimizer":"adam","histories":["val_loss","val_acc","loss","acc"],"seed":42,
↪"training_data":{},"csv_file":null,"meta_file":null,"use_model_name":"Full-Django-
↪AntiNex-Simple-Scaler-DNN","dataset":"/opt/antinex/antinex-datasets/v1/webapps/
↪django/training-ready/v1_django_cleaned.csv","predict_rows":null,"apply_scaler
↪":true,"predict_feature":"label_value","features_to_process":["idx","arp_hwlen",
↪"arp_hwtype","arp_id","arp_op","arp_plen","arp_ptype","dns_default_aa","dns_default_
↪ad","dns_default_an","dns_default_ancount","dns_default_ar","dns_default_arcount",
↪"dns_default_cd","dns_default_id","dns_default_length","dns_default_ns","dns_
↪default_nscount","dns_default_opcode","dns_default_qd","dns_default_qdcount","dns_
↪default_qr","dns_default_ra","dns_default_rcode","dns_default_rd","dns_default_tc",
↪"dns_default_z","dns_id","eth_id","eth_type","icmp_addr_mask","icmp_code","icmp_gw",
↪"icmp_id","icmp_ptr","icmp_seq","icmp_ts_ori","icmp_ts_rx","icmp_ts_tx","icmp_type",
↪"icmp_unused","ip_id","ip_ihl","ip_len","ip_tos","ip_version","ipv6_fl","ipv6_hlim",
↪"ipv6_nh","ipv6_plen","ipv6_tc","ipv6_version","ipvsix_id","pad_id","tcp_dport",
↪"tcp_fields_options.MSS","tcp_fields_options.NOP","tcp_fields_options.SAckOK","tcp_
↪fields_options.Timestamp","tcp_fields_options.WScale","tcp_id","tcp_seq","tcp_sport
↪","udp_dport","udp_id","udp_len","udp_sport"],"ignore_features":[],"sort_values":[],
↪"model_desc":{"layers":[{"num_neurons":200,"init":"uniform","activation":"relu"},{
↪"num_neurons":1,"init":"uniform","activation":"sigmoid"}]},"label_rules":{"labels":[
↪"not_attack","not_attack","attack"],"label_values":[-1,0,1]},"post_proc_rules":null,
↪"model_weights_file":"/tmp/ml_weights_job_4_result_4.h5","verbose":1,"version":1,
↪"publish_to_core":true,"worker_result_node":{"source":"drf","auth_url":"redis://
↪localhost:6379/9","ssl_options":{},"exchange":"drf_network_pipeline.pipeline.tasks.
↪task_ml_process_results","exchange_type":"topic","routing_key":"drf_network_
↪pipeline.pipeline.tasks.task_ml_process_results","queue":"drf_network_pipeline.
↪pipeline.tasks.task_ml_process_results","delivery_mode":2,"task_name":"drf_network_
↪pipeline.pipeline.tasks.task_ml_process_results","manifest":{"job_id":4,"result_id
↪":4,"job_type":"train-and-predict"}}},"training_data":{},"pre_proc":{},"post_proc":
↪{},"meta_data":{},"tracking_id":"ml_cb723821-e840-45ad-ac3a-94a86a0cfc88","version
↪":1,"created":"2018-03-30 16:32:56","updated":"2018-03-30 16:32:56","deleted":""},
↪"results":{"id":4,"user_id":1,"user_name":"root","job_id":4,"status":"initial",
↪"test_size":0.2,"csv_file":null,"meta_file":null,"version":1,"acc_data":{"accuracy
↪":-1.0},"error_data":null,"model_json":null,"model_weights":null,"acc_image_file
↪":null,"predictions_json":null,"created":"2018-03-30 16:32:56","updated":"2018-03-
↪30 16:32:56","deleted":""}}
```

4.5.5 Get New Predictions Results from the Pre-trained Deep Neural Network with Curl

Get the example Deep Neural Network Training, Accuracy and Prediction Results by ID **4**.

Note: This will return all **30200** records so it can take a second

```
auth_header="Authorization: JWT ${token}"
curl -s -X GET \
```

(continues on next page)

(continued from previous page)

```
    --header 'Content-Type: application/json' \
    --header 'Accept: application/json' \
    --header "${auth_header}" \
    'http://0.0.0.0:8010/mlresults/4/'

...


{"id":4,"user_id":1,"user_name":"root","job_id":4,"status":"finished","test_size":0.2,
↪"csv_file":null,"meta_file":null,"version":1,"acc_data":{"accuracy":99.
↪82615894039735},"error_data":null,"model_json":"{\"class_name\": \"Sequential\", \
↪"config\": [{\"class_name\": \"Dense\", \"config\": {\"name\": \"dense_1\", \
↪"trainable\": true, \"batch_input_shape\": [null, 67], \"dtype\": \"float32\", \
↪"units\": 200, \"activation\": \"relu\", \"use_bias\": true, \"kernel_initializer\
↪": {\"class_name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\":␣
↪0.05, \"seed\": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\
↪": {}}, \"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_
↪regularizer\": null, \"kernel_constraint\": null, \"bias_constraint\": null}}, {\
↪"class_name\": \"Dense\", \"config\": {\"name\": \"dense_2\", \"trainable\": true, \
↪"units\": 1, \"activation\": \"sigmoid\", \"use_bias\": true, \"kernel_initializer\
↪": {\"class_name\": \"RandomUniform\", \"config\": {\"minval\": -0.05, \"maxval\":␣
↪0.05, \"seed\": null}}, \"bias_initializer\": {\"class_name\": \"Zeros\", \"config\
↪": {}}, \"kernel_regularizer\": null, \"bias_regularizer\": null, \"activity_
↪regularizer\": null, \"kernel_constraint\": null, \"bias_constraint\": null}}], \
↪"keras_version\": \"2.1.5\", \"backend\": \"tensorflow\"}","model_weights":{},"acc_
↪image_file":null,"predictions_json":{"predictions":[

... lots of prediction dictionaries
```

4.5.6 Using Python Scripts

There are many example scripts through the repositories that use python to interface with each of the AntiNex components.

Here are some links to additional, more up-to-date examples:

https://github.com/jay-johnson/train-ai-with-django-swagger-jwt#automation

https://github.com/jay-johnson/antinex-core#publish-a-predict-request

https://github.com/jay-johnson/antinex-client#run-predictions

4.6 Debugging

4.6.1 Tail the API logs

From the base directory of the Django REST API repository you can watch what the server is doing inside the container with:

```
./tail-api.sh
```

4.6.2 Tail the Celery Worker logs

From the base directory of the Django REST API repository you can watch what the REST API Celery Worker is doing inside the container with:

```
./tail-worker.sh
```

4.6.3 Tail the AntiNex Core Worker logs

From the base directory of the Django REST API repository you can watch what the AntiNex Core Worker (which is also a Celery Worker) is doing inside the container with:

```
./tail-core.sh
```

4.6.4 Signature has expired

Log back in to get a new token if you see this message:

```
{"detail":"Signature has expired."}
```

```
token=$(curl -s -X POST \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
-d '{ "username": "root", "password": "123321" }' \
'http://0.0.0.0:8010/api-token-auth/' \
| sed -e 's/"/ /g' | awk '{print $4}')
```

4.7 AntiNex Stack Status

The AntiNex REST API is part of the AntiNex stack:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

CHAPTER 5

More Included App URLs

5.1 Jupyter Slides on How the Analysis Works

Note: The **left** and **right** arrow keys navigate the slides in the browser.

http://localhost:8889/Slides-AntiNex-Protecting-Django.slides.html#/

http://localhost:8890/Slides-AntiNex-Using-Pre-Trained-Deep-Neural-Networks-For-Defense.slides.html#/

5.2 Django REST API with Swagger

Credentials: **root** and **123321**

http://localhost:8010/swagger/

- Build and Train a DNN
- Get Training Predictions and Accuracy Results
- Get Training Job Record
- Prepare a New Dataset

5.3 Django-hosted Sphinx Docs

http://localhost:8010/docs/

5.4 Jupyter

Login with: **admin**

http://localhost:8888/

5.5 Browse the Postgres DB with pgAdmin4

Credentials: **admin@email.com** and **postgres**

http://localhost:83

CHAPTER 6

So why does this matter?

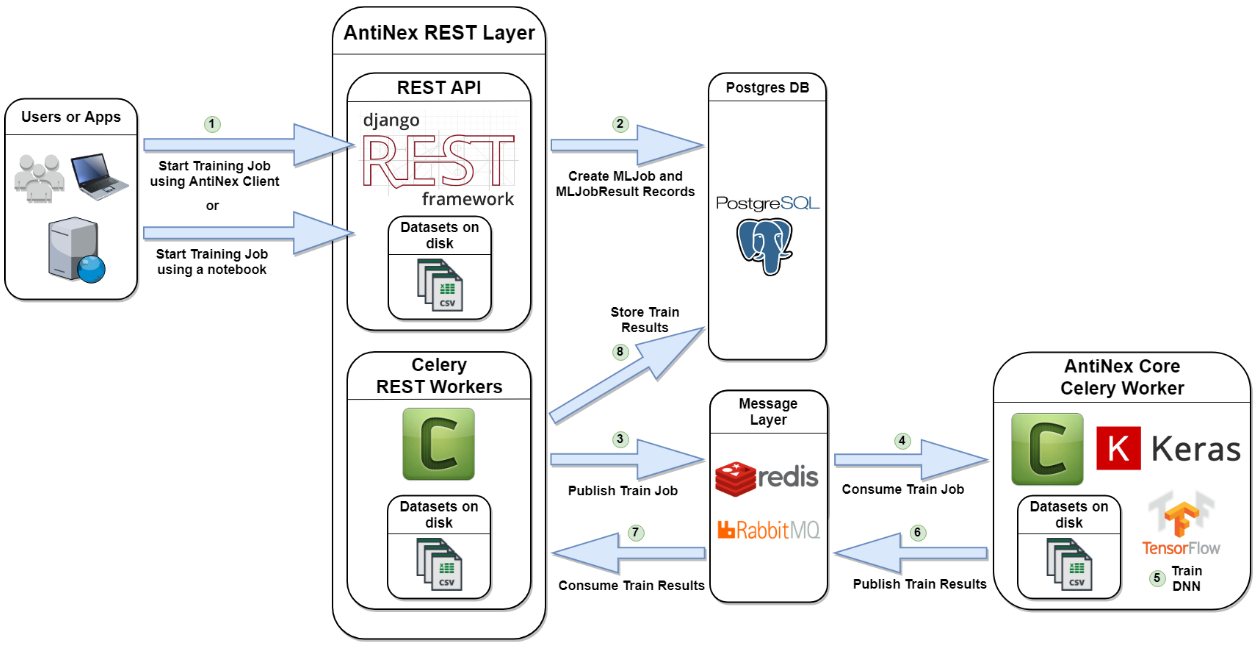
- There is no free software we can use today that can share and continually learn how to better defend software applications and our networks against attacks
- AI for network security is a vendor lock-in play, and this approach is already beating the best scores I see online
- Without open datasets and shared best-of-AI-model definitions, our networks will continue to be susceptible to attacks that are easy to defend (antivirus has been doing this same approach for years but it is not good enough)
- Build your own 99.7% accurate dnn within minutes of running the dockerized stack
- Building new training datasets with your own attack and non-attack data takes a matter of minutes
- Replay and prediction history is stored on the user's account within the included postgres database
- The same core can run on any system that can run python 3 (it can be backported to python 2 for IoT devices as all the internal components like Keras and Tensorflow still run on python 2)

CHAPTER 7

How does it work?

AntiNex is three custom python components that run distributed and are independently scalable. Like many other distributed systems, it utilizes a publisher-subscriber implementation to run a data pipeline with the final step being everything gets recorded in the postgres database (including all training, predictions and model definitions).

Here is the workflow for training of a Deep Neural Network with AntiNex. As a user you just have to start the docker stack, and submit a request over HTTP:



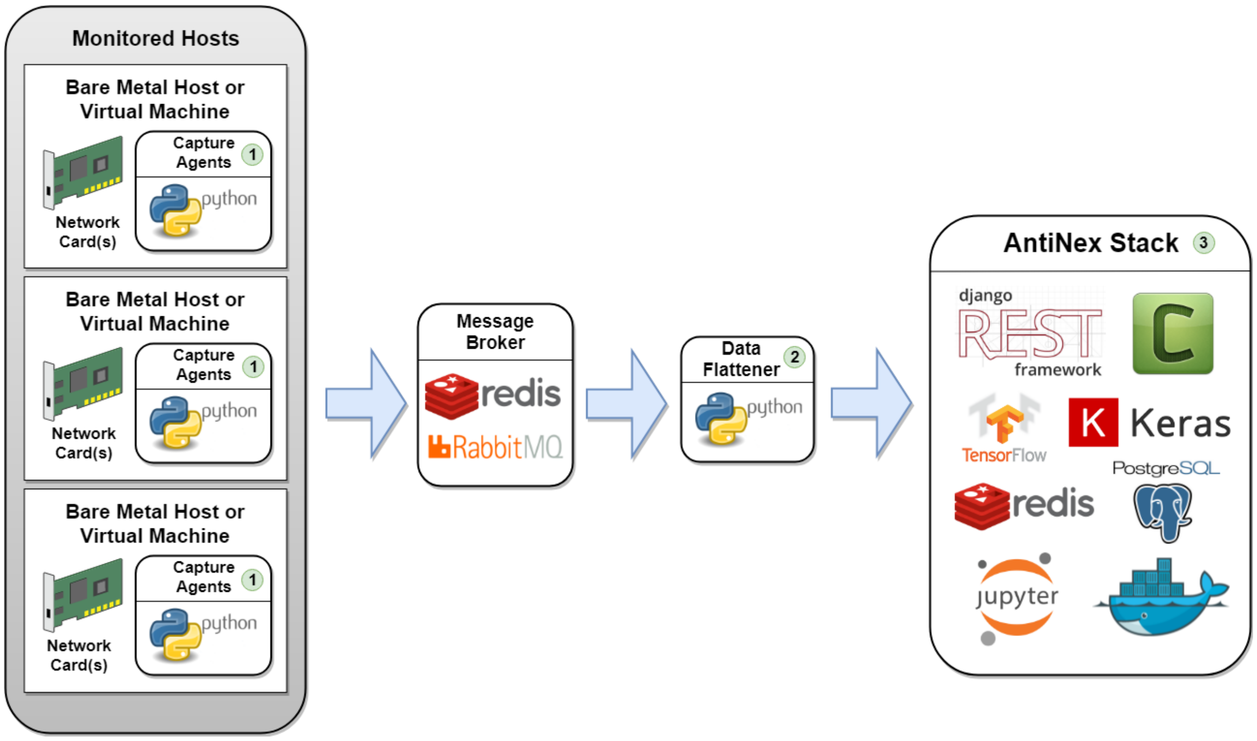
CHAPTER 8

Components

8.1 Network Pipeline

Traffic Capture Data Pipeline

Here is how the **capture agents** would be set up for capturing network traffic across many hosts. These agents create a network traffic feed that is aggregated in a common, shared message broker (redis by default).



Warning: **Capture agents** are going to sniff your network so be extremely careful where you deploy them. **Capture agents** must be run as **root** to capture traffic from all OSI layers. Also, **capture agents** should not run inside docker containers as docker is very noisy on the network (which I did not know when I started building this). Lastly, none of the docker compose files should be monitoring your network traffic without your explicit knowledge. Please contact me if you find one that does, and I will immediately remove it.

Note: The included **pipeline** container is only running the subscriber that saves CSVs and POSTs predictions to the REST API which make it easier to get started. Run this to verify what is running in the container:

```
docker exec -it pipeline ps auwwx
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
runner       1  0.2  0.0   4340   812 ?        Ss   07:32   0:00 /bin/sh -c cd /opt/
↪antinex/pipeline && . ~/.venvs/venvdrfpipeline/bin/activate && /opt/antinex/
↪pipeline/network_pipeline/scripts/packets_redis.py
runner      10 12.8  0.5 409060 66196 ?        Sl   07:32   0:00 python /opt/antinex/
↪pipeline/network_pipeline/scripts/packets_redis.py
runner      17  0.0  0.0  19192  2408 pts/0    Rs+  07:32   0:00 ps auwwx
```

This subscriber script is on GitHub:

https://github.com/jay-johnson/network-pipeline/blob/master/network_pipeline/scripts/packets_redis.py

Please refer to the repository for the latest code and documentation: https://github.com/jay-johnson/network-pipeline

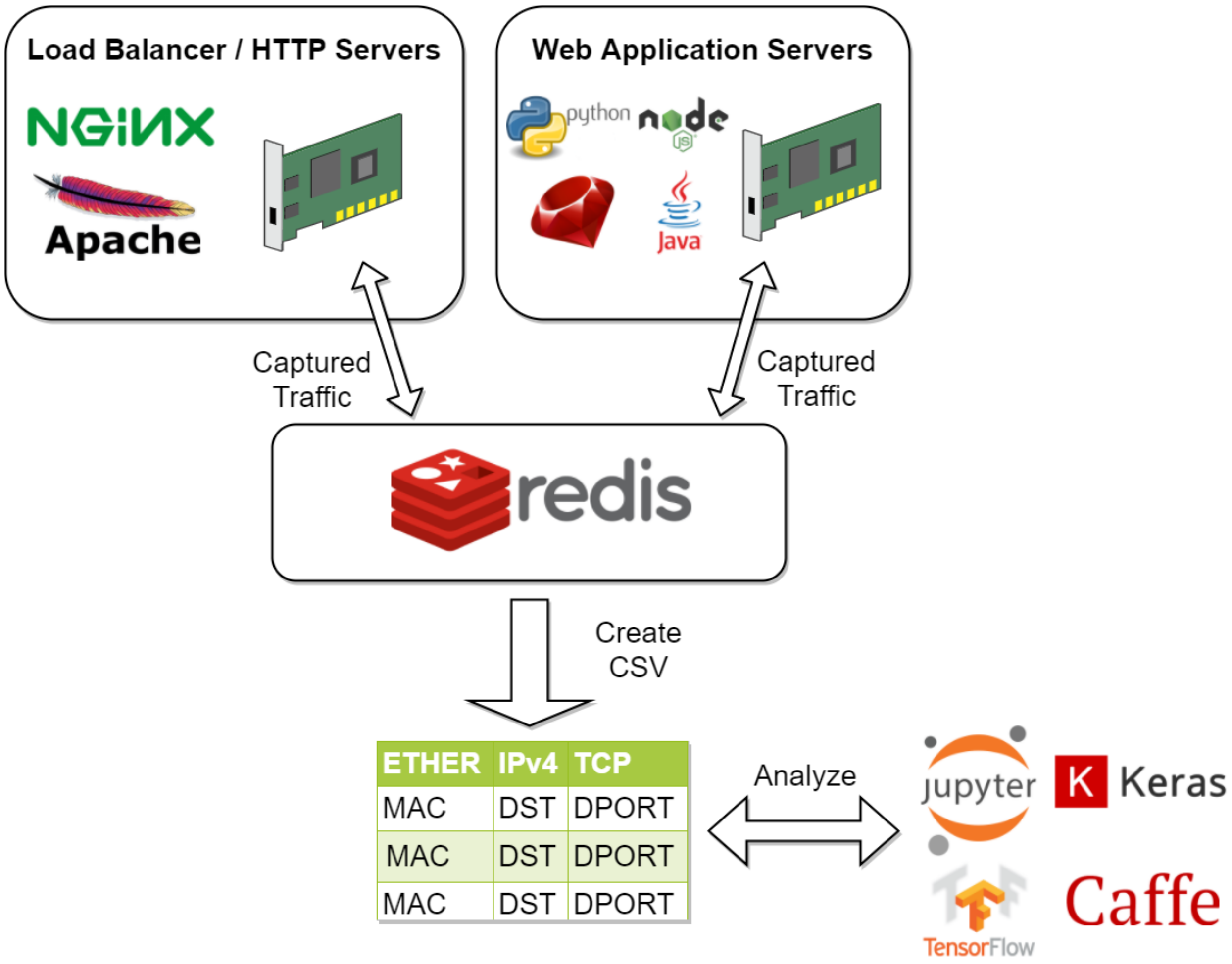
This repository allows users to capture network traffic in real-time from many machines running any of the capture agents. These agents become a network traffic feed which is aggregated in a common hub (redis).

These pre-configured capture agents perform the following steps in order:

1. Record network traffic based off easy-to-write network filters
2. Flatten captured traffic packets into dictionaries (using pandas json-normalize)
   (a) Assemble a csv file after capturing a configurable number of packets (100 by default)
   (b) Save the csv data to disk
3. Post the csv data as JSON to the REST API using the antinex-client

8.1.1 AntiNex - Network Data Analysis Pipeline

This is a distributed python 3 framework for automating network traffic capture and converting it into a csv file. Once you have a csv file you can build, train and tune machine learning models to defend your own infrastructure by actively monitoring the network layer.



It supports auto-publishing captured network traffic to the AntiNex REST API for using pre-trained Deep Neural Networks to make predictions on if this is an attack record or not using the AntiNex Core. Please refer to the Making Live Predictions using Pre-trained Neural Networks section for more details. Publishing to the REST API can run inside docker as well.

There are many choices to build a machine learning or AI model but for now I am using Jupyter Hub to build a pre-trained model for defending against OWASP Dynamic Analysis tools for finding vulnerabilities running in my owasp-jenkins repository.

- Django REST Framework + JWT + Swagger - run **prepare-dataset** and **train-keras-deep-neural-network** using a multi-tenant Django 2.0+ REST API server supporting JWT and Swagger
- Simulations directory - capturing simulated attacks using ZAP with Django, Flask, React, Vue, and Spring
- Prepare Dataset section - preparing training csvs from captured recordings
- Train Models section - training machine learning and AI models from prepared csvs and please check out the AntiNex Core which has accuracies over 99.8% and a Jupyter notebook
- Datasets repository - captured recordings if you want to see what some of the data will look like

8.1.2 Why?

After digging into how Internet Chemotherapy worked with a simple Nerfball approach, I wanted to see if I could train machine learning and AI models to defend this type of attack. Since the network is the first line to defend on the edge, on-premise or in the cloud, I wanted to start building the first line of defense and open source it. Also I do not know of any other toolchains to build defensive models using the network layer for free.

This repository automates dataset creation for training models by capturing network traffic on layers 2, 3 and 4 of the OSI model. Once a dataset has been Prepared it can be used to Train a Deep Neural Network. Pre-trained Deep Neural Networks can make live predictions on good or bad network traffic with the AntiNex Core.

8.1.3 How does it work?

This framework uses free open source tools to create the following publish-subscriber workflow:

1. Network traffic matches a capture tool filter
2. Capture tool converts packet layers into JSON
3. Capture tool publishes converted JSON dictionary to a message broker (Redis or RabbitMQ)
4. Packet processor consumes dictionary from message broker
5. Packet processor flattens dictionary
6. Packet processor periodically writes csv dataset from collected, flattened dictionaries (configurable for snapshotting csv on n-th number of packets consumed)
7. Flatten packets are published using JWT to a pre-trained Deep Neural Network for making predictions on if the network traffic is good or bad

Envisioned Deployment

- For on-premise and cloud environments, this framework would deploy capture tools to load balancers and application servers. These capture tool agents would publish to a redis cluster outside of the load balancers and application servers for analysis. By doing this, models could also be tuned to defend on the load balancer tier or application server tier independently.
- Remote edge machines would be running deployed, pre-trained, package-maintained models that are integrated with a prediction API. Periodic uploads of new, unexpected records would be sent encrypted back to the cloud for retraining models for helping defend an IoT fleet.

Detailed Version

The pipeline is a capture forwarding system focused on redundancy and scalability. Components-wise there are pre-configured capture tools that hook into the network devices on the operating system. If the capture tools find any traffic that matches their respective filter, then they json-ify the captured packet and forward it as a nested dictionary to a redis server (rabbitmq works as well, but requires setting the environment variables for authentication). Once the traffic packet dictionaries are in redis/rabbitmq, the packet processor consumes the nested dictionary and flattens them using pandas. The packet processors are set up to write csv datasets from the consumed, flattened dictionaries every 100 packets (you can configure the `SAVE_AFTER_NUM` environment variable to a larger number too).

Here are the included, standalone capture tools (all of which require root privileges to work):

1. capture_arp.py
2. capture_icmp.py
3. capture_ssh.py
4. capture_tcp.py
5. capture_telnet.py

6. capture_udp.py

AntiNex Stack Status

AntiNex Network Pipeline is part of the AntiNex stack:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

8.1.4 What packets and layers are supported?

Layer 2

- Ethernet
- ARP

Layer 3

- IPv4
- IPv6
- ICMP

Layer 4

- TCP
- UDP
- Raw - hex data from TCP or UDP packet body

Layer 5

- DNS

How do I get started?

1. Install from pypi or build the development environment

```
pip install network-pipeline
```

Or you can set up the repository locally

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/network-pipeline.git /opt/antinex/
↪pipeline
cd /opt/antinex/pipeline
virtualenv -p python3 /tmp/netpipevenv && source /tmp/netpipevenv/bin/activate &&␣
↪pip install -e .
```

2. Start Redis

   This guide assumes redis is running in docker, but as long as there's an accessible redis server on port 6379 you can use that too. RabbitMQ works as well, but requires setting the environment variables for connectivity.

```
# if you do not have docker-compose installed, you can try to install it with:
# pip install docker-compose
./start.sh
```

3. Verify Redis is Working

```
redis-cli
```

   or

```
telnet localhost 6379
```

4. Start Packet Processor for Consuming Messages

   Activate the virtual environment

```
source /tmp/netpipevenv/bin/activate
```

   Start it up

```
./network_pipeline/scripts/packets_redis.py
```

8.1.5 Making Live Predictions using Pre-trained Neural Networks

There are a few ways to make live predictions depending on how the pipeline and AntiNex assets are deployed:

1. Running the Full Django REST API stack using compose.yml (Co-located mode)

   This will start the Packet Processor using the default compose.yml file:

   https://github.com/jay-johnson/train-ai-with-django-swagger-jwt/blob/0d280216e3697f0d2cf7456095e37df64be73040/compose.yml#L105

   Clone the repo:

```
git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /
↪opt/antinex/api
cd /opt/antinex/api
```

   Start the co-located container stack with the `compose.yml` file:

```
docker-compose -f compose.yml up -d
```

2. Running Only the Network Pipeline compose.yml (Distributed mode)

   This will just start the Network Pipeline container and assumes the REST API is running on another host.

https://github.com/jay-johnson/network-pipeline/blob/master/compose.yml

Use the command:

```
docker-compose -f compose.yml up
```

3. Running the Packet Processor Manually Using Environment Variables (Development mode)

   Make sure to source the correct environment file before running `packets_redis.py` (Packet Processor).

   As an example the repository has a version that works with the compose.yml docker deployment:

```
source envs/antinex-dev.env
```

   When building your own credentials and datasets, you may have special characters in the env file. Please use `set -o allexport; source envs/antinex-dev.env; set +o allexport;` to handle this case.

   Right now the defaults do not have special characters, so the `source` command works just fine:

```
export ANTINEX_PUBLISH_ENABLED=1
export ANTINEX_URL=http://localhost:8010
export ANTINEX_USER=root
export ANTINEX_EMAIL=123321
export ANTINEX_PASSWORD=123321
export ANTINEX_PUBLISH_TO_CORE=1
export ANTINEX_USE_MODEL_NAME=Full-Django-AntiNex-Simple-Scaler-DNN
export ANTINEX_PUBLISH_REQUEST_FILE=/opt/antinex/client/examples/predict-rows-
↪scaler-full-django.json
export ANTINEX_FEATURES_TO_PROCESS=idx,arp_hwlen,arp_hwtype,arp_id,arp_op,arp_
↪plen,arp_ptype,dns_default_aa,dns_default_ad,dns_default_an,dns_default_ancount,
↪dns_default_ar,dns_default_arcount,dns_default_cd,dns_default_id,dns_default_
↪length,dns_default_ns,dns_default_nscount,dns_default_opcode,dns_default_qd,dns_
↪default_qdcount,dns_default_qr,dns_default_ra,dns_default_rcode,dns_default_rd,
↪dns_default_tc,dns_default_z,dns_id,eth_id,eth_type,icmp_addr_mask,icmp_code,
↪icmp_gw,icmp_id,icmp_ptr,icmp_seq,icmp_ts_ori,icmp_ts_rx,icmp_ts_tx,icmp_type,
↪icmp_unused,ip_id,ip_ihl,ip_len,ip_tos,ip_version,ipv6_fl,ipv6_hlim,ipv6_nh,
↪ipv6_plen,ipv6_tc,ipv6_version,ipvsix_id,pad_id,tcp_dport,tcp_fields_options.
↪MSS,tcp_fields_options.NOP,tcp_fields_options.SAckOK,tcp_fields_options.
↪Timestamp,tcp_fields_options.WScale,tcp_id,tcp_seq,tcp_sport,udp_dport,udp_id,
↪udp_len,udp_sport
export ANTINEX_IGNORE_FEATURES=
export ANTINEX_SORT_VALUES=
export ANTINEX_ML_TYPE=classification
export ANTINEX_PREDICT_FEATURE=label_value
export ANTINEX_SEED=42
export ANTINEX_TEST_SIZE=0.2
export ANTINEX_BATCH_SIZE=32
export ANTINEX_EPOCHS=15
export ANTINEX_NUM_SPLITS=2
export ANTINEX_LOSS=binary_crossentropy
export ANTINEX_OPTIMIZER=adam
export ANTINEX_METRICS=accuracy
export ANTINEX_HISTORIES=val_loss,val_acc,loss,acc
export ANTINEX_VERSION=1
export ANTINEX_CONVERT_DATA=1
export ANTINEX_CONVERT_DATA_TYPE=float
export ANTINEX_MISSING_VALUE=-1.0
export ANTINEX_INCLUDE_FAILED_CONVERSIONS=false
```

(continues on next page)

(continued from previous page)

```
export ANTINEX_CLIENT_VERBOSE=1
export ANTINEX_CLIENT_DEBUG=0
```

Load the Deep Neural Network into the AntiNex Core

Note: If you are running without the docker containers, please make sure to clone the client and datasets to disk:

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/antinex-client.git /opt/antinex/client
git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/antinex-
↪datasets
```

Load the Django Model into the Core

Please note this can take a couple minutes…

```
ai_train_dnn.py -u root -p 123321 -f deep-neural-networks/full-django.json

...

30196    -1.0 -1.000000  -1.000000
30197    -1.0 -1.000000  -1.000000
30198    -1.0 -1.000000  -1.000000
30199    -1.0 -1.000000  -1.000000

[30200 rows x 72 columns]
```

8.1.6 Capture Network Traffic

These tools are installed with the pip and require running with root to be able to hook into the local network devices for capturing traffic correctly.

Scapy currently provides the traffic capture tooling, but the code already has a semi-functional scalable, multi-processing engine to replace it. This will be ideal for dropping on a heavily utilized load balancer tier and run as an agent managed as a systemd service.

1. Login as root

   ```
   sudo su
   ```

2. Activate the Virtual Environment

   ```
   source /tmp/netpipevenv/bin/activate
   ```

3. Capture TCP Data

   By default TCP capture is only capturing traffic on ports: 80, 443, 8010, and 8443. This can be modified with the `NETWORK_FILTER` environment variable. Please avoid capturing on the redis port (default 6379) and rabbitmq port (default 5672) to prevent duplicate sniffing on the already-captured data that is being forwarded to the message queues which are ideally running in another virtual machine.

   This guide assumes you are running all these tools from the base directory of the repository.

```
./network_pipeline/scripts/capture_tcp.py
```

Capture SSH Traffic

```
./network_pipeline/scripts/capture_ssh.py
```

Capture Telnet Traffic

```
./network_pipeline/scripts/capture_telnet.py
```

4. Capture UDP Data

   With another terminal, you can capture UDP traffic at the same time

   ```
   sudo su
   ```

   Start UDP capture tool

   ```
   source /tmp/netpipevenv/bin/activate && ./network_pipeline/scripts/capture_udp.py
   ```

5. Capture ARP Data

   With another terminal, you can capture ARP traffic at the same time

   ```
   sudo su
   ```

   Start ARP capture tool

   ```
   source /tmp/netpipevenv/bin/activate && ./network_pipeline/scripts/capture_arp.py
   ```

6. Capture ICMP Data

   With another terminal, you can capture ICMP traffic at the same time

   ```
   sudo su
   ```

   Start ICMP capture tool

   ```
   source /tmp/netpipevenv/bin/activate && ./network_pipeline/scripts/capture_icmp.py
   ```

8.1.7 Simulating Network Traffic

ZAP Testing with Web Applications

The repository includes ZAPv2 simulations targeting the follow application servers:

- Django 2.0.1
- Flask RESTplus with Swagger
- React + Redux

- Vue
- Spring Pet Clinic

I will be updating this guide with more ZAP simulation tests in the future.

Please refer to the Simulations README for more details on running these to capture network traffic during an attack.

Quick Simulations

If you want to just get started, here are some commands and tools to start simulating network traffic for seeding your csv datasets.

1. Send a TCP message

```
./network_pipeline/scripts/tcp_send_msg.py
```

2. Send a UDP message

   (Optional) Start a UDP server for echo-ing a response on port 17000

```
sudo ./network_pipeline/scripts/listen_udp_port.py
2018-01-27T17:39:47.725377 - Starting UDP Server address=127.0.0.1:17000␣
↪backlog=5 size=1024 sleep=0.5 shutdown=/tmp/udp-shutdown-listen-server-127.0.0.
↪1-17000
```

   Send the UDP message

```
./network_pipeline/scripts/udp_send_msg.py
sending UDP: address=('0.0.0.0', 17000) msg=testing UDP msg time=2018-01-27␣
↪17:40:04 - cc9cdc1a-a900-48c5-acc9-b8ff5919087b
```

   (Optional) Verify the UDP server received the message

```
2018-01-27T17:40:04.915469 received UDP data=testing UDP msg time=2018-01-27␣
↪17:40:04 - cc9cdc1a-a900-48c5-acc9-b8ff5919087b
```

3. Simulate traffic with common shell tools

```
nslookup 127.0.0.1; nslookup 0.0.0.0; nslookup localhost
```

```
dig www.google.com; dig www.cnn.com; dig amazon.com
```

```
wget https://www.google.com; wget http://www.cnn.com; wget https://amazon.com
```

```
ping google.com; ping amazon.com
```

4. Run all of them at once

```
nslookup 127.0.0.1; nslookup 0.0.0.0; nslookup localhost; dig www.google.com; dig␣
↪www.cnn.com; dig amazon.com; wget https://www.google.com; wget http://www.cnn.
↪com; wget https://amazon.com; ping google.com; ping amazon.com
```

Capturing an API Simulation

Simulations that can automate + fuzz authenticated REST API service layers like ZAP are available in the AntiNex datasets repository for training Deep Neural Networks. The included Flask ZAP Simulation does login using OAuth

2.0 with ZAP for REST API validation, but there is a known issue with the swagger openapi integration within ZAP that limits the functionality (for now):

https://github.com/zaproxy/zaproxy/issues/4072

1. Start a local server listening on TCP port 80

```
sudo ./network_pipeline/scripts/listen_tcp_port.py
2018-01-27T23:59:22.344687 - Starting Server address=127.0.0.1:80 backlog=5␣
↪size=1024 sleep=0.5 shutdown=/tmp/shutdown-listen-server-127.0.0.1-80
```

2. Run a POST curl

```
curl -i -vvvv -POST http://localhost:80/TESTURLENDPOINT -d '{"user_id", "1234",
↪"api_key": "abcd", "api_secret": "xyz"}'
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> POST /TESTURLENDPOINT HTTP/1.1
> Host: localhost
> User-Agent: curl/7.55.1
> Accept: */*
> Content-Length: 59
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 59 out of 59 bytes
POST /TESTURLENDPOINT HTTP/1.1
Host: localhost
User-Agent: curl/7.55.1
Accept: */*
Content-Length: 59
Content-Type: application/x-www-form-urlencoded

* Connection #0 to host localhost left intact
{"user_id", "1234", "api_key": "abcd", "api_secret": "xyz"}
```

3. Verify local TCP server received the POST

```
2018-01-28T00:00:54.445294 received msg=7 data=POST /TESTURLENDPOINT HTTP/1.1
Host: localhost
User-Agent: curl/7.55.1
Accept: */*
Content-Length: 59
Content-Type: application/x-www-form-urlencoded

{"user_id", "1234", "api_key": "abcd", "api_secret": "xyz"} replying
```

Larger Traffic Testing

1. Host a local server listening on TCP port 80 using `nc`

```
sudo nc -l 80
```

2. Send a large TCP msg to the `nc` server

```
./network_pipeline/scripts/tcp_send_large_msg.py
```

8.1.8 Inspecting the CSV Datasets

By default, the dataset csv files are saved to: `/tmp/netdata-*.csv` and you can set a custom path by exporting the environment variables `DS_NAME`, `DS_DIR` or `OUTPUT_CSV` as needed.

```
ls /tmp/netdata-*.csv
/tmp/netdata-2018-01-27-13-13-58.csv  /tmp/netdata-2018-01-27-13-18-25.csv  /tmp/
↪netdata-2018-01-27-16-44-08.csv
/tmp/netdata-2018-01-27-13-16-38.csv  /tmp/netdata-2018-01-27-13-19-46.csv
/tmp/netdata-2018-01-27-13-18-03.csv  /tmp/netdata-2018-01-27-13-26-34.csv
```

8.1.9 Prepare Dataset

This is a guide for building training datasets from the recorded csvs in the network pipeline datasets repository. Once a dataset is prepared locally, you can use the modelers to build and tune machine learning and AI models.

Install

This will make sure your virtual environment is using the latest `pandas` pip and install the latest ML/AI pips. Please run it from the repository's base directory.

```
source /tmp/netpipevenv/bin/activate
pip install --upgrade -r ./network_pipeline/scripts/builders/requirements.txt
```

Overview

I have not uploaded a local recording from my development stacks, so for now this will prepare a training dataset by randomly applying `non-attack - 0` and `attack - 1` labels for flagging records as **attack** and **non-attack** records.

Setup

Please export the path to the datasets repository on your host:

```
export DS_DIR=<path_to_datasets_base_directory>
```

Or clone the repository to the default value for the environment variable (`DS_DIR=/opt/antinex/datasets`) with:

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/network-pipeline-datasets.git /opt/antinex/
↪datasets
```

Build Dataset

This will take a few moments to prepare the csv files.

```
prepare_dataset.py
2018-01-31 23:38:04,298 - builder - INFO - start - builder
2018-01-31 23:38:04,298 - builder - INFO - finding pipeline csvs in dir=/opt/antinex/
↪datasets/*/*.csv
2018-01-31 23:38:04,299 - builder - INFO - adding file=/opt/antinex/datasets/react-
↪redux/netdata-2018-01-29-13-36-35.csv
2018-01-31 23:38:04,299 - builder - INFO - adding file=/opt/antinex/datasets/spring/
↪netdata-2018-01-29-15-00-12.csv
2018-01-31 23:38:04,299 - builder - INFO - adding file=/opt/antinex/datasets/vue/
↪netdata-2018-01-29-14-12-44.csv
2018-01-31 23:38:04,299 - builder - INFO - adding file=/opt/antinex/datasets/django/
↪netdata-2018-01-28-23-12-13.csv
2018-01-31 23:38:04,299 - builder - INFO - adding file=/opt/antinex/datasets/django/
↪netdata-2018-01-28-23-06-05.csv
2018-01-31 23:38:04,299 - builder - INFO - adding file=/opt/antinex/datasets/flask-
↪restplus/netdata-2018-01-29-11-30-02.csv
```

Verify Dataset and Tracking Files

By default the environment variable `OUTPUT_DIR` writes the dataset csv files to `/tmp`:

```
ls -lrth /tmp/*.csv
-rw-rw-r-- 1 jay jay  26M Jan 31 23:38 /tmp/fulldata_attack_scans.csv
-rw-rw-r-- 1 jay jay 143K Jan 31 23:38 /tmp/cleaned_attack_scans.csv
```

Additionally, there are data governance, metadata and tracking files created as well:

```
ls -lrth /tmp/*.json
-rw-rw-r-- 1 jay jay 2.7K Jan 31 23:38 /tmp/fulldata_metadata.json
-rw-rw-r-- 1 jay jay 1.8K Jan 31 23:38 /tmp/cleaned_metadata.json
```

8.1.10 Train Models

I am using Keras to train a Deep Neural Network to predict **attack (1)** and **non-attack (0)** records using a prepared dataset. Please checkout the keras_dnn.py module if you are interested in learning more. Please let me know if there are better ways to set up the neural network layers or hyperparameters as well.

1. Source the virtual environment

   ```
   source /tmp/netpipevenv/bin/activate
   ```

2. (Optional) Train with a different dataset

   By default the environment variable `CSV_FILE=/tmp/cleaned_attack_scans.csv` can be changed to train models with another prepared dataset.

   To do so run:

   ```
   export CSV_FILE=<path_to_csv_dataset_file>
   ```

8.1.11 Train a Keras Deep Neural Network

Included in the pip is a `keras_dnn.py` script. Below is a sample log from a training run that scored an **83.33%** accuracy predicting **attack** vs **non-attack** records.

Please note, this can take a few minutes if you are not using a GPU. Also the accuracy results will be different depending on how you set up the model.

```
keras_dnn.py
Using TensorFlow backend.
2018-02-01 00:01:30,653 - keras-dnn - INFO - start - keras-dnn
2018-02-01 00:01:30,653 - keras-dnn - INFO - Loading csv=/tmp/cleaned_attack_scans.csv
2018-02-01 00:01:30,662 - keras-dnn - INFO - Predicting=label_value with features=[
↪'eth_type', 'idx', 'ip_ihl', 'ip_len', 'ip_tos', 'ip_version', 'label_value', 'tcp_
↪dport', 'tcp_fields_options.MSS', 'tcp_fields_options.Timestamp', 'tcp_fields_
↪options.WScale', 'tcp_seq', 'tcp_sport'] ignore_features=['label_name', 'ip_src',
↪'ip_dst', 'eth_src', 'eth_dst', 'src_file', 'raw_id', 'raw_load', 'raw_hex_load',
↪'raw_hex_field_load', 'pad_load', 'eth_dst', 'eth_src', 'ip_dst', 'ip_src']␣
↪records=2217
2018-02-01 00:01:30,664 - keras-dnn - INFO - splitting rows=2217 into X_train=1773 X_
↪test=444 Y_train=1773 Y_test=444
2018-02-01 00:01:30,664 - keras-dnn - INFO - creating sequential model
2018-02-01 00:01:30,705 - keras-dnn - INFO - compiling model
2018-02-01 00:01:30,740 - keras-dnn - INFO - fitting model - please wait
Train on 1773 samples, validate on 444 samples
Epoch 1/50
2018-02-01 00:01:30.947551: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your␣
↪CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.
↪1 SSE4.2 AVX AVX2
1773/1773 [==============================] - 1s 704us/step - loss: 2.5727 - acc: 0.
↪8404 - val_loss: 2.6863 - val_acc: 0.8333
Epoch 2/50
1773/1773 [==============================] - 1s 626us/step - loss: 2.5727 - acc: 0.
↪8404 - val_loss: 2.6863 - val_acc: 0.8333

...

Epoch 50/50
1773/1773 [==============================] - 1s 629us/step - loss: 2.5727 - acc: 0.
↪8404 - val_loss: 2.6863 - val_acc: 0.8333
444/444 [==============================] - 0s 17us/step
2018-02-01 00:02:29,118 - keras-dnn - INFO - Accuracy: 83.33333333333334
```

Optional Tweaks

1. Colorized Logging for Debugging

   Export the path to the colorized logger config. This examples assumes you are in the base directory of the repository.

   ```
   export LOG_CFG=$(pwd)/network_pipeline/log/colors-logging.json
   ```

Linting

flake8 .

pycodestyle –exclude=./simulations,.tox,.eggs

License

Apache 2.0 - Please refer to the LICENSE for more details

8.2 REST API

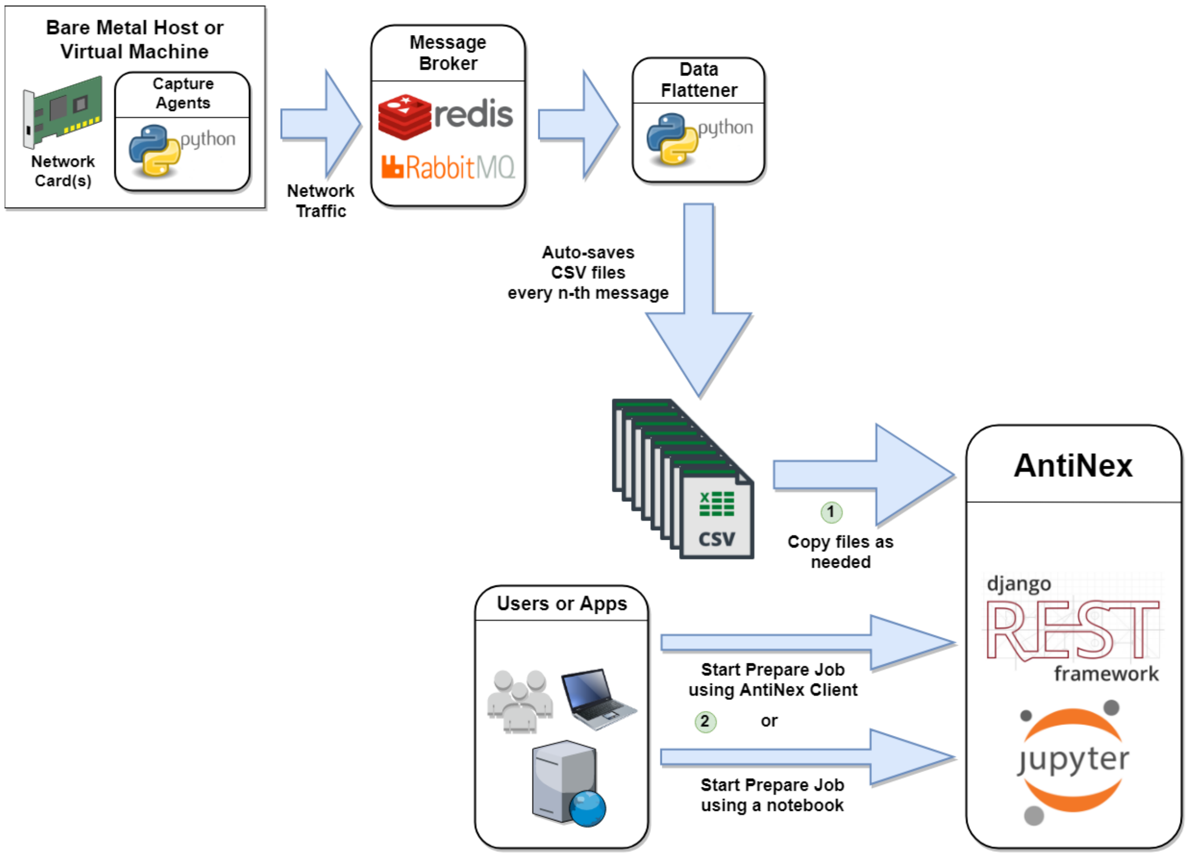
Multi-tenant service with Swagger, Celery and JWT

Please refer to the repository for the latest code and documentation: https://github.com/jay-johnson/train-ai-with-django-swagger-jwt

The REST API is the gateway for running anything in AntiNex. Only authenticated users can use the included API requests for:

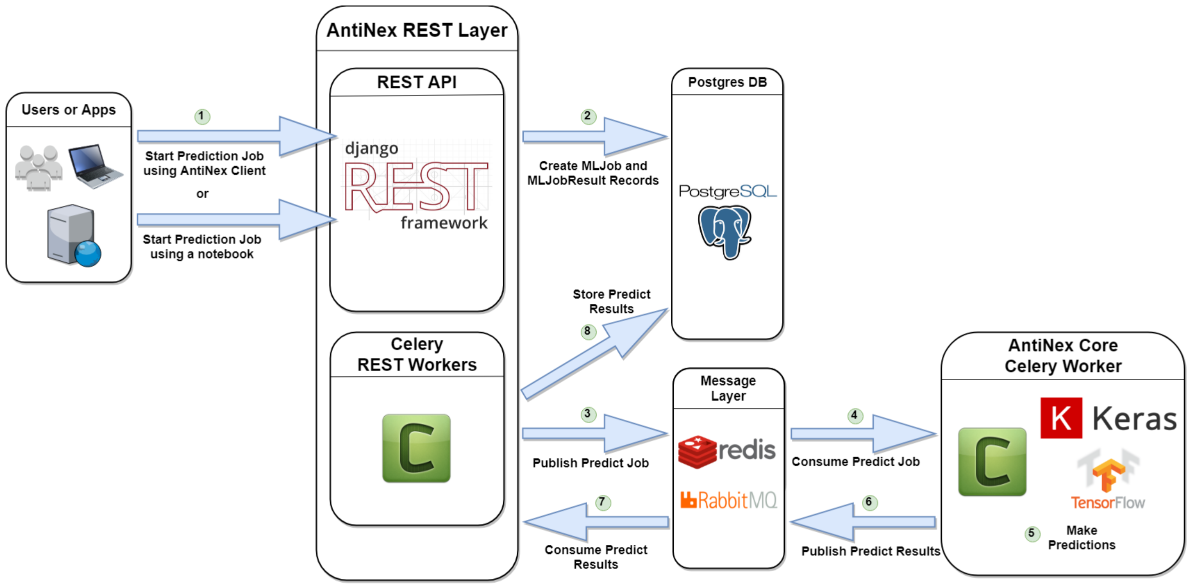
1. Preparing a New Dataset

   Here is the workflow for Preparing Datasets. CSVs must be synced across the hosts running the REST API and Celery Workers to function.



2. Running a Training Job
3. Making New Predictions using Pre-trained Deep Neural Networks

   Here is the workflow. Notice CSVs are not required on any of the hosts anymore.



4. Getting a Job's record
5. Getting a Job's Results including predictions
6. Managing User accounts

8.2.1 AntiNex REST API

Automate training AI to defend applications with a Django 2.0+ REST Framework + Celery + Swagger + JWT using Keras and Tensorflow.

Now supports building the same highly accurate deep neural networks as the AntiNex Core (**99.8%** accuracy with Django, Flask, React + Redux, Vue and Spring). This repository is fully dockerized and after the django celery worker finishes processing, it will auto-push predictions to the core's celery worker which is decoupled from django and the django database. The core's celery worker stores pre-trained AI neural networks in memory for faster predictions and supports re-training models as needed.

For those wanting to scale up their processing speeds, AntiNex deploys on OpenShift Container Platform and Kubernetes with persistent database volumes for Postgres (Crunchy Data) and Redis (Bitnami)

AntiNex Stack Status

The AntiNex REST API is part of the AntiNex stack:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

Supported API Requests

- Prepare a Dataset
- Train a Deep Neural Network from a Prepared Dataset using Keras and Tensorflow
- Multi-Tenant Deep Neural Network Training with Simulations
- Get recent Training jobs (including Models as json and weights)
- Get recent Training results (nice for reviewing historical accuracy)
- Get recent Prepared Datasets
- Creating and managing users

This repository was built to help capture `non-attack` network traffic and to improve the accuracy of the Keras + Tensorflow Deep Neural Networks by providing them a simple multi-tenant REST API that has Swagger + JWT authentication baked into a single web application. By default, all created Deep Neural Networks are automatically saved as JSON including model weights. It also does not require a database (unless you want to set it up), and will be scaled out with Celery Connectors in the future. Please refer to the Network Pipeline repository for more details. This Django application server also comes with a functional Celery worker for running heavyweight, time-intensive tasks required for asynchronous use cases. This is good for when you are trying to train a deep net that takes a few minutes, and you do not want your HTTP client to time out.

I plan to automate the tests in a loop and then release the captured HTTP traffic to compile the first `non-attack` dataset for pairing up with the OWASP `attack` data which is already recorded and available in:

https://github.com/jay-johnson/network-pipeline-datasets

Update: 2018-02-25 - These merged datasets and accuracies are now available in the repository:

https://github.com/jay-johnson/antinex-datasets

8.2.2 Watch Getting Started

Assuming your host has the pips already cached locally this takes about a minute.

8.2.3 Install

Tested on Ubuntu 17.10, Ubuntu 18.04 and works on OpenShift Container Platform with Kubernetes.

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /opt/
↪antinex/api
cd /opt/antinex/api
./install.sh
```

8.2.4 Getting Started With Docker

You can run without these optional steps and just use the default SQLite database. If you want to use docker and download all the containers, you can use the `compose.yml` file to start all of the containers and download the latest `ai-core` docker image which is ~2.5 GB on disk (built with Dockerfile and stored on Docker Hub).

To start all run:

```
# if you do not have docker compose installed, you can try installing it with:
# pip install docker-compose
./run-all.sh
```

Verify the containers started

```
docker ps
CONTAINER ID        IMAGE                                    COMMAND                    ␣
↪CREATED             STATUS              PORTS                     NAMES
d34c8973066b        jayjohnson/antinex-pipeline:latest       "/bin/sh -c 'cd /opt..."   2␣
↪hours ago          Up 2 hours                                    pipeline
12ef5482bc17        jayjohnson/antinex-worker:latest         "/bin/sh -c 'cd /opt..."   2␣
↪hours ago          Up 2 hours                                    worker
da7970ae165f        jayjohnson/antinex-api:latest            "/bin/sh -c 'cd /opt..."   2␣
↪hours ago          Up 2 hours                                    api
11a2c95b7247        jayjohnson/antinex-core:latest           "/bin/sh -c 'cd /opt..."   2␣
↪hours ago          Up 2 hours                                    core
1f26d89c8c2c        jayjohnson/antinex-jupyter:latest        "/opt/antinex/core/d..."   2␣
↪hours ago          Up 2 hours                                    jupyter
4905682ff3b4        postgres:10.4-alpine                      "docker-entrypoint.s..."   2␣
↪hours ago          Up 2 hours          0.0.0.0:5432->5432/tcp   postgres
fd8300740935        redis:4.0.9-alpine                        "docker-entrypoint.s..."   2␣
↪hours ago          Up 2 hours          0.0.0.0:6379->6379/tcp   redis
7c682ba78adb        jayjohnson/pgadmin4:1.0.0                 "python ./usr/local/..."   2␣
↪hours ago          Up 2 hours          0.0.0.0:83->5050/tcp      pgadmin
```

Quick links

If you are running all the containers, you can use these links to move around:

- Use Swagger to Train a new Deep Neural Network (login with `trex` and `123321`)

  http://localhost:8010/swagger/#!/ml/ml_create

- Jupyter Notebook showing how the Deep Neural Networks are Trained (login with `admin` and `ALT + r` to view the slideshow)

  http://localhost:8888/notebooks/AntiNex-Protecting-Django.ipynb

- Jupyter Notebook shoing how to use Pre-trained Deep Neural Networks with AntiNex

  http://localhost:8888/notebooks/AntiNex-Using-Pre-Trained-Deep-Neural-Networks-For-Defense.ipynb

If you are interested in running locally without the large container image, you can run the broker and database stack with docker containers for simulating a more production-ready environment. Here's the containers these steps will start:

1. Postgres 10
2. Redis (Pub/Sub, Caching and Celery Tasks)
3. pgAdmin4 - Web app for managing Postgres

Here's how to run it:

1. Source the environment

   ```
   source envs/drf-dev.env
   ```

2. Start the Stack

```
./run-stack.sh
Starting stack: full-stack-dev.yml
Creating postgres ... done
Creating pgadmin ...
Creating postgres ...
```

3. Verify the containers are running

```
docker ps
CONTAINER ID        IMAGE                           COMMAND                  CREATED ␣
↪           STATUS              PORTS                                                   ␣
↪                                                         NAMES
2c7cfbd9328e        postgres:10.2-alpine            "docker-entrypoint.s..."   3␣
↪minutes ago        Up 3 minutes        0.0.0.0:5432->5432/tcp                          ␣
↪                                                                postgres
9c34c9588349        jayjohnson/pgadmin4:1.0.0       "python ./usr/local/..."   3␣
↪minutes ago        Up 3 minutes        0.0.0.0:83->5050/tcp                            ␣
↪                                                                pgadmin
75e325113424        redis:4.0.5-alpine              "docker-entrypoint.s..."   3␣
↪minutes ago        Up 3 minutes        0.0.0.0:6379->6379/tcp                          ␣
↪                                                                redis
```

4. Initialize the Postgres database

```
export USE_ENV=drf-dev
./run-migrations.sh
```

5. Login to pgAdmin4

   http://localhost:83/browser/

   User: `admin@email.com` Password: `postgres`

6. Register the Postgres server

   (a) Right click on "Servers" and then "Create Server"

   (b) On the "General" tab enter a name like "webapp"

   (c) On the "Connection" tab enter:

      Host: postgres

      Username: postgres

      Password: postgres

   (d) Click "Save password?" check box

   (e) Click the "Save" button

   (f) Navigate down the tree:

      Servers > webapp (or the name you entered) > Databases > webapp > Schemas > public > Tables

   (g) Confirm there's database tables with names like:

```
pipeline_mljob
pipeline_mljobresult
pipeline_mlprepare
```

8.2.5 Start

By default, this project uses gunicorn to start, but you can change to uwsgi by running `export APP_SERVER=uwsgi` before starting. Both app servers should work just fine.

Note: if you are running the docker "full stack" please make sure to run: `export USE_ENV=drf-dev` before starting the django application, or you can use `run-django.sh` which should do the same as `start.sh`.

```
./start.sh

Starting Django listening on TCP port 8010
http://localhost:8010/swagger

[2018-02-07 11:27:20 -0800] [10418] [INFO] Starting gunicorn 19.7.1
[2018-02-07 11:27:20 -0800] [10418] [INFO] Listening at: http://127.0.0.1:8010 (10418)
[2018-02-07 11:27:20 -0800] [10418] [INFO] Using worker: sync
[2018-02-07 11:27:20 -0800] [10418] [INFO] DJANGO_DEBUG=yes - auto-reload enabled
[2018-02-07 11:27:20 -0800] [10418] [INFO] Server is ready. Spawning workers
[2018-02-07 11:27:20 -0800] [10422] [INFO] Booting worker with pid: 10422
[2018-02-07 11:27:20 -0800] [10422] [INFO] Worker spawned (pid: 10422)
[2018-02-07 11:27:20 -0800] [10423] [INFO] Booting worker with pid: 10423
[2018-02-07 11:27:20 -0800] [10423] [INFO] Worker spawned (pid: 10423)
[2018-02-07 11:27:20 -0800] [10424] [INFO] Booting worker with pid: 10424
[2018-02-07 11:27:20 -0800] [10424] [INFO] Worker spawned (pid: 10424)
[2018-02-07 11:27:20 -0800] [10426] [INFO] Booting worker with pid: 10426
[2018-02-07 11:27:20 -0800] [10426] [INFO] Worker spawned (pid: 10426)
[2018-02-07 11:27:20 -0800] [10430] [INFO] Booting worker with pid: 10430
[2018-02-07 11:27:20 -0800] [10430] [INFO] Worker spawned (pid: 10430)
```

8.2.6 Celery Worker

Start the Worker

Start the Celery worker in a new terminal to process published Django work tasks for heavyweight, time-intensive operations.

```
./run-worker.sh
```

Create User

Create the user `trex` with password `123321`:

```
source tests/users/user_1.sh \
&& ./tests/create-user.sh \
&& env | grep API | sort

Creating user: trex on http://localhost:8010/users/
{"id":2,"username":"trex","email":"bugs@antinex.com"}
Getting token for user: trex
{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJ1c2VyX2lkIjo2LCJ1c2VybmFtZSI6InRyZXgiLCJleHAiOjE1MjgyNjExMjgsImVtYWlsIjoiYnVnc0BhbnRpbmV4LmNvbS
↪W6Lb2N1v8S3e6EMT7RuTvfUQMTbKjrmYzhMxtFQ9jhk"}
API_DEBUG=false
API_EMAIL=bugs@antinex.com
```

(continues on next page)

(continued from previous page)

```
API_FIRSTNAME=Guest
API_LASTNAME=Guest
API_PASSWORD=123321
API_URL=http://localhost:8010
API_USER=trex
API_VERBOSE=true
```

8.2.7 Automation

All of these scripts run in the `tests` directory:

```
cd tests
```

Make sure the virtual environment has been loaded:

```
source ~/.venvs/venvdrfpipeline/bin/activate
```

Clone the datasets repository

git clone https://github.com/jay-johnson/network-pipeline-datasets /opt/antinex/datasets

Prepare a new Dataset from Captured Recordings

```
./build-new-dataset.py
```

Train a Keras Deep Neural Network with Tensorflow

```
./create-keras-dnn.py

...

2018-02-03 00:31:24,342 - create-keras-dnn - INFO - SUCCESS - Post Response␣
↪status=200 reason=OK
2018-02-03 00:31:24,342 - create-keras-dnn - INFO - {'job': {'id': 1, 'user_id': 2,
↪'user_name': 'trex', 'title': 'Keras DNN - network-pipeline==1.0.9', 'desc':
↪'Tensorflow backend with simulated data', 'ds_name': 'cleaned', 'algo_name': 'dnn',
↪'ml_type': 'keras', 'status': 'initial', 'control_state': 'active', 'predict_feature
↪': 'label_value', 'training_data': {}, 'pre_proc': {}, 'post_proc': {}, 'meta_data
↪': {}, 'tracking_id': 'ml_701552d5-c761-4c69-9258-00d05ff81a48', 'version': 1,
↪'created': '2018-02-03 08:31:17', 'updated': '2018-02-03 08:31:17', 'deleted': ''},
↪'results': {'id': 1, 'user_id': 2, 'user_name': 'trex', 'job_id': 1, 'status':
↪'finished', 'version': 1, 'acc_data': {'accuracy': 83.7837837300859}, 'error_data':␣
↪None, 'created': '2018-02-03 08:31:24', 'updated': '2018-02-03 08:31:24', 'deleted
↪': ''}}
```

Create a Highly Accurate Deep Neural Network for Protecting Django

This is the same API request the core uses to build the Django DNN with an accuracy of **99.8%**:

https://github.com/jay-johnson/antinex-core#accuracy-and-prediction-report with Notebook:

https://github.com/jay-johnson/antinex-core/blob/master/docker/notebooks/AntiNex-Protecting-Django.ipynb

```
./create-keras-dnn.py -f ./scaler-full-django-antinex-simple.json

Please wait... this can take a few minutes

...

2018-03-21 06:04:48,314 - ml_tasks - INFO - saving job=83 results
2018-03-21 06:04:50,387 - ml_tasks - INFO - updating job=83 results=83
2018-03-21 06:04:53,957 - ml_tasks - INFO - task - ml_job - done - ml_job.id=83 ml_
↪result.id=83 accuracy=99.81788079470199 predictions=30200
```

Train and Predict with just a Dictionary List of Records

This will send a list of records to the API to train and make predictions. This mimics the live-prediction capability in the core for reusing pre-trained DNNs to make predictions faster. I use it to send the newest records to predict, so I do not have to generate lots of csv files everywhere + all-the-time.

```
./create-keras-dnn.py -f ./predict-rows-scaler-full-django.json
```

Train and Predict using the AntiNex Core

This will train and cache a deep neural network using the AntiNex Core. Once trained, the core can make future predictions with the same API call without having to retrain. This makes predictions much faster.

```
./create-keras-dnn.py -f only-publish-scaler-full-django.json
```

The core trains a deep neural network and persists it in a dictionary that uses the label value on the request to store the trained model. Future predictions must continue to reuse the same `label` value on the request to avoid waiting for a retraining cycle. Here is the label value used in the previous request which is:

```
"label": "Full-Django-AntiNex-Simple-Scaler-DNN"
```

Make Predictions for a List of Records

If you have a list of records the API, Worker and Core support making predictions for each record in a list.

Predict using the AntiNex Worker:

```
./create-keras-dnn.py -f predict-rows-scaler-full-django.json
```

Predict using the AntiNex Core:

```
./create-keras-dnn.py -f only-publish-predict-rows-simple.json
```

8.2.8 Advanced Naming for Multi-Tenant Environments

Problems will happen if multiple users are sharing the same host's `/tmp/` directory with the default naming conventions. To prevent issues, it is recommended to change the output dataset directory to separate directories per user and to make sure the directories are accessible by the Django server processes. Here's an example of changing the output directory to my user which triggers the custom name detection. This detection means I will see logs for the training command to run with my newly generated dataset and metadata files:

```
mkdir /opt/jay
export OUTPUT_DIR=/opt/jay/
./build-new-dataset.py

...

Train a Neural Network with:
./create-keras-dnn.py /opt/jay/cleaned_attack_scans.csv /opt/jay/cleaned_metadata.json
```

If changing the output directory is not possible, then users will need to make sure the file names are unique before running. Here's an example naming strategy for the csv datasets and metadata files to prevent collisions. The `build-new-dataset.py` script will also suggest the training command to run when you activate custom names:

Prepare a Named Dataset

```
./build-new-dataset.py /tmp/<MyFirstName>_$(date +"%Y-%m-%d-%H-%m-%N")_full.csv /tmp/
↪<MyFirstName>_$(date +"%Y-%m-%d-%H-%m-%N")_readytouse.csv
```

Example that shows the suggested training command to run using the named dataset files on disk:

```
./build-new-dataset.py /tmp/jay_$(date +"%Y-%m-%d-%H-%m-%N")_full.csv /tmp/jay_$(date␣
↪+"%Y-%m-%d-%H-%m-%N")_readytouse.csv

...

Train a Neural Network with:
./create-keras-dnn.py /tmp/jay_2018-02-05-21-02-274468596_readytouse.csv /tmp/cleaned_
↪meta-54525d8da8a54e9d9005a29c63f2918b.json
```

Confirm the files were created:

```
ls -lrth /tmp/jay_2018-02-05-21-02-274468596_readytouse.csv /tmp/cleaned_meta-
↪54525d8da8a54e9d9005a29c63f2918b.json
-rw-rw-r-- 1 jay jay 143K Feb  5 21:23 /tmp/jay_2018-02-05-21-02-274468596_readytouse.
↪csv
-rw-rw-r-- 1 jay jay 1.8K Feb  5 21:23 /tmp/cleaned_meta-
↪54525d8da8a54e9d9005a29c63f2918b.json
```

Please note, if you use filenames and set the `OUTPUT_DIR` environment variable, the environment variable takes priority (even if you specify `/path/to/some/dir/uniquename.csv`). The dataset and metadata files will be stored in the `OUTPUT_DIR` directory:

```
echo $OUTPUT_DIR
/opt/jay/

./build-new-dataset.py jay_$(date +"%Y-%m-%d-%H-%m-%N")_full.csv jay_$(date +"%Y-%m-
↪%d-%H-%m-%N")_readytouse.csv

...

Train a Neural Network with:
./create-keras-dnn.py /opt/jay/jay_2018-02-05-22-02-521671337_readytouse.csv /opt/jay/
↪cleaned_meta-2b961845162a4d6e9e382c6f540302fe.json
```

(continues on next page)

(continued from previous page)

8.2.9 Swagger

Create a User

http://localhost:8010/swagger/#!/users/users_create

Click on the yellow `Example Value` section to paste in defaults or paste in your version of:

```
{
    "username": "trex",
    "password": "123321",
    "email": "bugs@antinex.com"
}
```

Login User

If you want to login as the super user:

- Username: `trex`
- Password: `123321`

http://localhost:8010/api-auth/login/

Logout User

http://localhost:8010/swagger/?next=/swagger/#!/accounts/accounts_logout_create

8.2.10 JWT

Get a Token

This will validate authentication with JWT is working:

```
./get_user_jwt_token.sh
{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJ1c2VyX2lkIjo0LCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MTc1OTg3NTIsImVtYWlsIjoicm9vdEBlbWFpbC5jb20ifQ
↪ip3Lj5o4SCK4TARlDuLyw-Dc6qMkt8xUx8WsQwIn2uo"}
```

(Optional) If you have `jq` installed:

```
./get_user_jwt_token.sh | jq
{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJ1c2VyX2lkIjo0LCJ1c2VybmFtZSI6InJvb3QiLCJleHAiOjE1MTc1OTg3NDEsImVtYWlsIjoicm9vdEBlbWFpbC5jb20ifQ
↪WAIatDGkeFJbH6LL_4rRQaAydZXcE8j0KK7dBnA2GJU"
}
```

http://localhost:8010/swagger/?next=/swagger/#!/ml/ml_run_create

8.2.11 Development

Swagger Prepare a new Dataset from Captured Recordings

http://localhost:8010/swagger/#!/mlprepare/mlprepare_create

Paste in the following values and click **Try it Out**:

```
{
    "title": "Prepare new Dataset from recordings",
    "desc": "",
    "ds_name": "new_recording",
    "full_file": "/tmp/fulldata_attack_scans.csv",
    "clean_file": "/tmp/cleaned_attack_scans.csv",
    "meta_suffix": "metadata.json",
    "output_dir": "/tmp/",
    "ds_dir": "/opt/antinex/datasets",
    "ds_glob_path": "/opt/antinex/datasets/*/*.csv",
    "pipeline_files": {
        "attack_files": []
    },
    "meta_data": {},
    "post_proc": {
        "drop_columns": [
            "src_file",
            "raw_id",
            "raw_load",
            "raw_hex_load",
            "raw_hex_field_load",
            "pad_load",
            "eth_dst",
            "eth_src",
            "ip_dst",
            "ip_src"
        ],
        "predict_feature": "label_name"
    },
    "label_rules": {
        "set_if_above": 85,
        "labels": [
            "not_attack",
            "attack"
        ],
        "label_values": [
            0,
            1
        ]
    },
    "version": 1
}
```

Swagger Train a Keras Deep Neural Network with Tensorflow

http://0.0.0.0:8010/swagger/#!/ml/ml_create

Paste in the following values and click **Try it Out**:

1. Build the Django DNN for Predicting Network Attacks

```
{
    "label": "Full-Django-AntiNex-Simple-Scaler-DNN",
    "dataset": "/opt/antinex/antinex-datasets/v1/webapps/django/training-ready/v1_
↪django_cleaned.csv",
    "ml_type": "classification",
    "predict_feature": "label_value",
    "features_to_process": [
        <list of comma separated column names>
    ],
    "ignore_features": [
        <optional list of comma separated column names>
    ],
    "sort_values": [
        <optional list of comma separated column names>
    ],
    "seed": 42,
    "test_size": 0.2,
    "batch_size": 32,
    "epochs": 15,
    "num_splits": 2,
    "loss": "binary_crossentropy",
    "optimizer": "adam",
    "metrics": [
        "accuracy"
    ],
    "histories": [
        "val_loss",
        "val_acc",
        "loss",
        "acc"
    ],
    "model_desc": {
        "layers": [
            {
                "num_neurons": 200,
                "init": "uniform",
                "activation": "relu"
            },
            {
                "num_neurons": 1,
                "init": "uniform",
                "activation": "sigmoid"
            }
        ]
    },
    "label_rules": {
        "labels": [
            "not_attack",
            "not_attack",
            "attack"
        ],
        "label_values": [
            -1,
            0,
            1
        ]
    },
```

(continues on next page)

(continued from previous page)

```
    "version": 1
}
```

2. Prototyping with a List of Records

   I use this script to convert a configurable number of records from the bottom of a csv file which helps build these type of prediction json files:

   https://github.com/jay-johnson/antinex-core/blob/master/antinex_core/scripts/convert_bottom_rows_to_json.py

```
./create-keras-dnn.py -f ./readme-predict-demo-1.json
```

   Here are the contents of `./tests/readme-predict-demo-1.json`

```
{
    "label": "Prediction-Model-Prototyping",
    "predict_rows": [
        {
            "_dataset_index": 1,
            "label_value": 1,
            "more_keys": 54.0
        },
        {
            "_dataset_index": 2,
            "label_value": 1,
            "more_keys": 24.0
        },
        {
            "_dataset_index": 2,
            "label_value": 0,
            "more_keys": 33.0
        }
    ],
    "ml_type": "classification",
    "predict_feature": "label_value",
    "features_to_process": [
        "more_keys"
    ],
    "ignore_features": [
    ],
    "sort_values": [
    ],
    "seed": 42,
    "test_size": 0.2,
    "batch_size": 32,
    "epochs": 15,
    "num_splits": 2,
    "loss": "binary_crossentropy",
    "optimizer": "adam",
    "metrics": [
        "accuracy"
    ],
    "histories": [
        "val_loss",
        "val_acc",
        "loss",
```

(continues on next page)

(continued from previous page)

```
        "acc"
    ],
    "model_desc": {
        "layers": [
            {
                "num_neurons": 200,
                "init": "uniform",
                "activation": "relu"
            },
            {
                "num_neurons": 1,
                "init": "uniform",
                "activation": "sigmoid"
            }
        ]
    },
    "label_rules": {
        "labels": [
            "not_attack",
            "not_attack",
            "attack"
        ],
        "label_values": [
            -1,
            0,
            1
        ]
    },
    "version": 1
}
```

3. Deprecated - Using just CSV files

```
{
    "csv_file": "/tmp/cleaned_attack_scans.csv",
    "meta_file": "/tmp/cleaned_metadata.json",
    "title": "Keras DNN - network-pipeline==1.0.9",
    "desc": "Tensorflow backend with simulated data",
    "ds_name": "cleaned",
    "algo_name": "dnn",
    "ml_type": "keras",
    "predict_feature": "label_value",
    "training_data": "{}",
    "pre_proc": "{}",
    "post_proc": "{}",
    "meta_data": "{}",
    "version": 1
}
```

Verify the Celery Worker Processes a Task without Django

I find the first time I integrate Celery + Django + Redis can be painful. So I try to validate Celery tasks work before connecting Celery to Django over a message broker (like Redis). Here is a test tool for helping debug this integration with the celery-loaders project. It's also nice not having to click through the browser to debug a new task.

1. Run the task test script

```
./run-celery-task.py -t drf_network_pipeline.users.tasks.task_get_user -f tests/
↪celery/task_get_user.json
2018-06-05 22:41:39,426 - run-celery-task - INFO - start - run-celery-task
2018-06-05 22:41:39,426 - run-celery-task - INFO - connecting Celery=run-celery-
↪task broker=redis://localhost:6379/9 backend=redis://localhost:6379/10 tasks=[
↪'drf_network_pipeline.users.tasks']
2018-06-05 22:41:39,427 - get_celery_app - INFO - creating celery app=run-celery-
↪task tasks=['drf_network_pipeline.users.tasks']
2018-06-05 22:41:39,470 - run-celery-task - INFO - app.broker_url=redis://
↪localhost:6379/9 calling task=drf_network_pipeline.users.tasks.task_get_user␣
↪data={'celery_enabled': True, 'cache_key': None, 'use_cache': False, 'data': {
↪'user_id': 2}}
2018-06-05 22:41:39,535 - run-celery-task - INFO - calling task=drf_network_
↪pipeline.users.tasks.task_get_user - started job_id=4931e1fc-3610-4259-8ccd-
↪5724a1c50c79
2018-06-05 22:41:39,549 - run-celery-task - INFO - calling task=drf_network_
↪pipeline.users.tasks.task_get_user - success job_id=4931e1fc-3610-4259-8ccd-
↪5724a1c50c79 task_result={'status': 0, 'err': '', 'task_name': '', 'data': {'id
↪': 2, 'username': 'trex', 'email': 'bugs@antinex.com'}, 'celery_enabled': True,
↪'use_cache': False, 'cache_key': None}
2018-06-05 22:41:39,549 - run-celery-task - INFO - end - run-celery-task
```

2. Verify the Celery Worker Processed the Task

   If Redis and Celery are working as expected, the logs should print something similar to the following:

```
2018-06-06 05:41:39,535 - celery.worker.strategy - INFO - Received task: drf_
↪network_pipeline.users.tasks.task_get_user[4931e1fc-3610-4259-8ccd-5724a1c50c79]
2018-06-06 05:41:39,537 - user_tasks - INFO - task - task_get_user - start req_
↪node={'celery_enabled': True, 'cache_key': None, 'use_cache': False, 'data': {
↪'user_id': 2}}
2018-06-06 05:41:39,537 - user_tasks - INFO - finding user=2 cache=False
2018-06-06 05:41:39,539 - celery.worker.request - DEBUG - Task accepted: drf_
↪network_pipeline.users.tasks.task_get_user[4931e1fc-3610-4259-8ccd-
↪5724a1c50c79] pid:26
2018-06-06 05:41:39,547 - user_tasks - INFO - found user.id=2 name=trex
2018-06-06 05:41:39,547 - user_tasks - INFO - task - task_get_user result={'status
↪': 0, 'err': '', 'task_name': '', 'data': {'id': 2, 'username': 'trex', 'email
↪': 'bugs@antinex.com'}, 'celery_enabled': True, 'use_cache': False, 'cache_key
↪': None} - done
2018-06-06 05:41:39,550 - celery.app.trace - INFO - Task drf_network_pipeline.
↪users.tasks.task_get_user[4931e1fc-3610-4259-8ccd-5724a1c50c79] succeeded in 0.
↪013342023004952352s: {'status': 0, 'err': '', 'task_name': '', 'data': {'id': 2,
↪ 'username': 'trex', 'email': 'bugs@antinex.com'}, 'celery_enabled': True, 'use_
↪cache': False, 'cache_key': None}
```

Additional Legacy Client API Tools

These tools and examples were created before the AntiNex Python Client was released. Please use that for official API examples.

8.2.12 Get a Prepared Dataset

```
export PREPARE_JOB_ID=1
./get-a-prepared-dataset.py
```

8.2.13 Get an ML Job

Any trained Keras Deep Neural Network models are saved as an `ML Job`.

```
export JOB_ID=1
./get-a-job.py
```

8.2.14 Get an ML Job Result

```
export JOB_RESULT_ID=1
./get-a-result.py
```

8.2.15 Get Recent Prepared Datasets

```
./get-recent-datasets.py
```

8.2.16 Get Recent ML Jobs

```
./get-recent-jobs.py
```

8.2.17 Get Recent ML Job Results

This is nice for reviewing historical accuracy as your tune your models.

```
./get-recent-results.py
```

Run Tests

The unit tests can be run:

```
./run-tests.sh

...

PASSED - unit tests
```

Or run a single test

```
source envs/dev.env; cd webapp; source ~/.venvs/venvdrfpipeline/bin/activate
python manage.py test drf_network_pipeline.tests.test_ml.MLJobTest
```

8.2.18 Multi-Tenant Simulations

Simulations run from the `./tests/` directory.

```
cd tests
```

Run the default `user1` simulation in a new terminal:

```
./run-user-sim.py
```

In a new terminal start `user2` simulation:

```
./run-user-sim.py user2
```

In a new terminal start `user3` simulation:

```
./run-user-sim.py user3
```

Want to check how many threads each process is using?

It appears that either Keras or Tensorflow are using quite a bit of threads behind the scenes. On Ubuntu you can view the number of threads used by `gunicorn` or `uwsgi` with these commands:

```
ps -o nlwp $(ps awuwx | grep django | grep -v grep | awk '{print $2}')
```

If you're running `uwsgi` instead of the `gunicorn` use:

```
ps -o nlwp $(ps awuwx | grep uwsgi | grep -v grep | awk '{print $2}')
```

8.2.19 Stop Full Stack

If you are running the "full stack", then you can run this command to stop the docker containers:

```
./stop-stack.sh
```

Testing

1. Set up the Testing Runtime and Environment Variables

   ```
   source ~/.venvs/venvdrfpipeline/bin/activate
   source ./envs/dev.env
   ```

2. Change to the `webapp` directory

   Tests need to run in the same directory as the `manage.py`

   ```
   cd webapp
   ```

3. Run all Tests

   ```
   python manage.py test
   ```

4. Run all Test Cases in a Test module

   ```
   python manage.py test drf_network_pipeline.tests.test_ml
   ```

5. Run a Single Test Case

   ```
   python manage.py test drf_network_pipeline.tests.test_ml.MLJobTest.test_ml_
   ↪predict_helper_works
   ```

or

```
python manage.py test drf_network_pipeline.tests.test_user.AccountsTest.test_
↪create_user_with_invalid_email
```

Linting

flake8 .

pycodestyle –exclude=.tox,.eggs,migrations

License

Apache 2.0 - Please refer to the LICENSE for more details

8.2.20 Citations and Included Works

Special thanks to these amazing projects for helping make this easier!

Original Django project template from

https://github.com/jpadilla/django-project-template

Django REST Framework

https://github.com/encode/django-rest-framework

Celery

http://www.celeryproject.org/

User Registration

https://github.com/szopu/django-rest-registration

Swagger for Django

https://github.com/marcgibbons/django-rest-swagger

JWT for Django REST

https://github.com/GetBlimp/django-rest-framework-jwt

Keras

https://github.com/keras-team/keras

Tensorflow

https://github.com/tensorflow

SQLite

https://www.sqlite.org/index.html

Gunicorn

http://docs.gunicorn.org/

uWSGI

https://uwsgi-docs.readthedocs.io/en/latest/

pgAdmin

https://www.pgadmin.org/

PostgreSQL

https://www.postgresql.org/

Django Cacheops

https://github.com/Suor/django-cacheops

8.3 AntiNex Core

A Celery Worker that can Train and use Pre-trained Models

Please refer to the repository for the latest code and documentation: https://github.com/jay-johnson/antinex-core

The core is a backend worker that supports two API requests:

1. Train a new model
2. Predict using a pre-trained model (if the model does not exist it will initiate a training job)

By default, the core can support up to 100 pre-trained dnn's for making predictions. Once predictions are finished, the core uses celery to call the REST API's celery worker to record the results in the postgres database. The core is decoupled from a database for keeping it fast and so it can run on constrained environments (IoT).

In the future the core will support loading the weights and model files from disk and out of S3, but that's for a future release.

8.3.1 AntiNex Core

Automating network exploit detection using highly accurate pre-trained deep neural networks.

As of 2018-03-12, the core can repeatedly predict attacks on Django, Flask, React + Redux, Vue, and Spring application servers by training using the pre-recorded AntiNex datasets with cross validation scores above **~99.8%** with automated scaler normalization.

Accuracy + Training + Cross Validation in a Jupyter Notebook

https://github.com/jay-johnson/antinex-core/blob/master/docker/notebooks/AntiNex-Protecting-Django.ipynb

Using a Pre-Trained Deep Neural Network in a Jupyter Notebook

https://github.com/jay-johnson/antinex-core/blob/master/docker/notebooks/AntiNex-Using-Pre-Trained-Deep-Neural-Networks-For-D
ipynb

Overview

The core is a Celery worker pool for processing training and prediction requests for deep neural networks to detect network exploits (Nex) using Keras and Tensorflow in near real-time. Internally each worker manages a buffer of pre-trained models identified by the `label` from the initial training request. Once trained, a model can be used for rapid prediction testing provided the same `label` name is used on the prediction request. Models can also be re-trained by using the training api with the same `label`. While the initial focus is on network exploits, the repository also includes mock stock data for demonstrating running a worker pool to quickly predict regression data (like stock prices) with many, pre-trained deep neural networks.

This repository is a standalone training and prediction worker pool that is decoupled from the AntiNex REST API:

https://github.com/jay-johnson/train-ai-with-django-swagger-jwt

AntiNex Stack Status

AntiNex Core Worker is part of the AntiNex stack:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

Install

pip install antinex-core

Optional for Generating Images

If you want to generate images please install `python3-tk` on Ubuntu.

```
sudo apt-get install python3-tk
```

Docker

Start the container for browsing with Jupyter:

```
# if you do not have docker compose installed, you can try installing it with:
# pip install docker-compose
cd docker
./start-stack.sh
```

Open Jupyter Notebook with Django Deep Neural Network Analysis

Default password is: `admin`

http://localhost:8888/notebooks/AntiNex-Protecting-Django.ipynb

View Notebook Presentation Slides

1. Use `Alt + r` inside the notebook
2. Use the non-vertical scolling url: http://localhost:8889/Slides-AntiNex-Protecting-Django.slides.html
3. Use the non-vertical scolling url: http://localhost:8890/Slides-AntiNex-Using-Pre-Trained-Deep-Neural-Networks-For-Defense.slides.html

Run

Please make sure redis is running and accessible before starting the core:

```
redis-cli
127.0.0.1:6379>
```

With redis running and the antinex-core pip installed in the python 3 runtime, use this command to start the core:

```
./run-antinex-core.sh
```

Or with celery:

```
celery worker -A antinex_core.antinex_worker -l DEBUG
```

Publish a Predict Request

To train and predict with the new automated scaler-normalized dataset with a 99.8% prediction accuracy for detecting attacks using a wide, two-layer deep neural network with the AntiNex datasets run the following steps.

Clone

Please make sure to clone the dataset repo to the pre-configured location:

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/antinex-
↪datasets
```

Django - Train and Predict

```
./antinex_core/scripts/publish_predict_request.py -f training/scaler-full-django-
↪antinex-simple.json
```

Flask - Train and Predict

```
./antinex_core/scripts/publish_predict_request.py -f training/scaler-full-flask-
↪antinex-simple.json
```

React and Redux - Train and Predict

```
./antinex_core/scripts/publish_predict_request.py -f training/scaler-full-react-redux-
↪antinex-simple.json
```

Vue - Train and Predict

```
./antinex_core/scripts/publish_predict_request.py -f training/scaler-full-vue-antinex-
↪simple.json
```

Spring - Train and Predict

```
./antinex_core/scripts/publish_predict_request.py -f training/scaler-full-spring-
↪antinex-simple.json
```

Accuracy and Prediction Report

After a few minutes the final report will be printed out like:

```
2018-03-11 23:35:00,944 - antinex-prc - INFO - sample=30178 - label_value=1.0␣
↪predicted=1 label=attack
2018-03-11 23:35:00,944 - antinex-prc - INFO - sample=30179 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,944 - antinex-prc - INFO - sample=30180 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,944 - antinex-prc - INFO - sample=30181 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,944 - antinex-prc - INFO - sample=30182 - label_value=-1.0␣
↪predicted=-1 label=not_attack
```

(continues on next page)

(continued from previous page)

```
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30183 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30184 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30185 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30186 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30187 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30188 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30189 - label_value=1.0␣
↪predicted=1 label=attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30190 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,945 - antinex-prc - INFO - sample=30191 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30192 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30193 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30194 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30195 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30196 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30197 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30198 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,946 - antinex-prc - INFO - sample=30199 - label_value=-1.0␣
↪predicted=-1 label=not_attack
2018-03-11 23:35:00,947 - antinex-prc - INFO - Full-Django-AntiNex-Simple-Scaler-DNN␣
↪made predictions=30200 found=30200 accuracy=99.84685430463577
2018-03-11 23:35:00,947 - antinex-prc - INFO - Full-Django-AntiNex-Simple-Scaler-DNN -
↪ saving model=full-django-antinex-simple-scaler-dnn
```

If you do not have the datasets cloned locally, you can use the included minimized dataset from the repo:

```
./antinex_core/scripts/publish_predict_request.py -f training/scaler-django-antinex-
↪simple.json
```

Publish a Train Request

```
./antinex_core/scripts/publish_train_request.py
```

Publish a Regression Prediction Request

```
./antinex_core/scripts/publish_regression_predict.py
```

JSON API

The AntiNex core manages a pool of workers that are subscribed to process tasks found in two queues (`webapp.train.requests` and `webapp.predict.requests`). Tasks are defined as JSON dictionaries and must have the following structure:

```
{
    "label": "Django-AntiNex-Simple-Scaler-DNN",
    "dataset": "./tests/datasets/classification/cleaned_attack_scans.csv",
    "apply_scaler": true,
    "ml_type": "classification",
    "predict_feature": "label_value",
    "features_to_process": [
        "eth_type",
        "idx",
        "ip_ihl",
        "ip_len",
        "ip_tos",
        "ip_version",
        "tcp_dport",
        "tcp_fields_options.MSS",
        "tcp_fields_options.Timestamp",
        "tcp_fields_options.WScale",
        "tcp_seq",
        "tcp_sport"
    ],
    "ignore_features": [
    ],
    "sort_values": [
    ],
    "seed": 42,
    "test_size": 0.2,
    "batch_size": 32,
    "epochs": 10,
    "num_splits": 2,
    "loss": "binary_crossentropy",
    "optimizer": "adam",
    "metrics": [
        "accuracy"
    ],
    "histories": [
        "val_loss",
        "val_acc",
        "loss",
        "acc"
    ],
    "model_desc": {
        "layers": [
            {
                "num_neurons": 250,
                "init": "uniform",
                "activation": "relu"
            },
            {
                "num_neurons": 1,
                "init": "uniform",
                "activation": "sigmoid"
            }
```

(continues on next page)

(continued from previous page)

```
        ]
    },
    "label_rules": {
        "labels": [
            "not_attack",
            "not_attack",
            "attack"
        ],
        "label_values": [
            -1,
            0,
            1
        ]
    },
    "version": 1
}
```

Regression prediction tasks are also supported, and here is an example from an included dataset with mock stock prices:

```
{
    "label": "Scaler-Close-Regression",
    "dataset": "./tests/datasets/regression/stock.csv",
    "apply_scaler": true,
    "ml_type": "regression",
    "predict_feature": "close",
    "features_to_process": [
        "high",
        "low",
        "open",
        "volume"
    ],
    "ignore_features": [
    ],
    "sort_values": [
    ],
    "seed": 7,
    "test_size": 0.2,
    "batch_size": 32,
    "epochs": 50,
    "num_splits": 2,
    "loss": "mse",
    "optimizer": "adam",
    "metrics": [
        "accuracy"
    ],
    "model_desc": {
        "layers": [
            {
                "activation": "relu",
                "init": "uniform",
                "num_neurons": 200
            },
            {
                "activation": null,
                "init": "uniform",
```

(continues on next page)

(continued from previous page)

```
                "num_neurons": 1
            }
        ]
    }
}
```

Development

```
virtualenv -p python3 ~/.venvs/antinexcore && source ~/.venvs/antinexcore/bin/
↪activate && pip install -e .
```

Testing

Run all

```
python setup.py test
```

Run a test case

```
python -m unittest tests.test_train.TestTrain.test_train_antinex_simple_success_
↪retrain
```

Linting

flake8 .

pycodestyle .

License

Apache 2.0 - Please refer to the LICENSE for more details

CHAPTER 9

Additional Components

9.1 AntiNex Client

Please refer to the repository for the latest code and documentation: https://github.com/jay-johnson/antinex-client

This repository is a python client for interacting with the REST API.

9.1.1 AntiNex Python Client

Python API Client for training deep neural networks with the REST API running

https://github.com/jay-johnson/train-ai-with-django-swagger-jwt

Install

pip install antinex-client

AntiNex Stack Status

AntiNex client is part of the AntiNex stack:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

9.1.2 Run Predictions

These examples use the default user `root` with password `123321`. It is advised to change this to your own user in the future.

Train a Deep Neural Network with a JSON List of Records

```
ai -u root -p 123321 -f examples/predict-rows-scaler-django-simple.json
```

Train a Deep Neural Network to Predict Attacks with the AntiNex Datasets

Please make sure the datasets are available to the REST API, Celery worker, and AntiNex Core worker. The datasets are already included in the docker container `ai-core` provided in the default `compose.yml` file:

https://github.com/jay-johnson/train-ai-with-django-swagger-jwt/blob/51f731860daf134ea2bd3b68468927c614c83ee5/compose.yml#L53-L104

If you're running outside docker make sure to clone the repo with:

```
git clone https://github.com/jay-johnson/antinex-datasets.git /opt/antinex/antinex-
↪datasets
```

Train the Django Defensive Deep Neural Network

Please wait as this will take a few minutes to return and convert the predictions to a pandas DataFrame.

```
ai -u root -p 123321 -f examples/scaler-full-django-antinex-simple.json

...

[30200 rows x 72 columns]
```

Using Pre-trained Neural Networks to make Predictions

The AntiNex Core manages pre-trained deep neural networks in memory. These can be used with the REST API by adding the `"publish_to_core": true` to a request while running with the REST API compose.yml docker containers running.

Run:

```
ai -u root -p 123321 -f examples/publish-to-core-scaler-full-django.json
```

Here is the diff between requests that will run using a pre-trained model and one that will train a new neural network:

```
antinex-client$ diff examples/publish-to-core-scaler-full-django.json examples/scaler-
↪full-django-antinex-simple.json
5d4
<     "publish_to_core": true,
antinex-client$
```

Prepare a Dataset

```
ai_prepare_dataset.py -u root -p 123321 -f examples/prepare-new-dataset.json
```

Get Job Record for a Deep Neural Network

Get a user's MLJob record by setting: `-i <MLJob.id>`

This include the model json or model description for the Keras DNN.

```
ai_get_job.py -u root -p 123321 -i 4
```

Get Predictions Results for a Deep Neural Network

Get a user's MLJobResult record by setting: `-i <MLJobResult.id>`

This includes predictions from the training or prediction job.

```
ai_get_results.py -u root -p 123321 -i 4
```

Get a Prepared Dataset

Get a user's MLPrepare record by setting: `-i <MLPrepare.id>`

```
ai_get_prepared_dataset.py -u root -p 123321 -i 15
```

Using a Client Built from Environment Variables

This is how the Network Pipeline streams data to the AntiNex Core to make predictions with pre-trained models.

Export the example environment file:

```
source examples/example-prediction.env
```

Run the client prediction stream script

```
ai_env_predict.py -f examples/predict-rows-scaler-full-django.json
```

Development

```
virtualenv -p python3 ~/.venvs/antinexclient && source ~/.venvs/antinexclient/bin/
↪activate && pip install -e .
```

Testing

Run all

```
python setup.py test
```

Linting

flake8 .

pycodestyle .

License

Apache 2.0 - Please refer to the LICENSE for more details

9.2 AntiNex Utils

Please refer to the repository for the latest code and documentation: https://github.com/jay-johnson/antinex-client

This repository is a standalone library that uses Scikit-Learn, Keras and Tensorflow to:

1. Create dnn's from either: JSON or default values
2. Transform datasets into scaler normalized values
3. Make predictions with new or pre-trained dnn's for classification and regression problems
4. Merge predictions with the original dataset for easier review and analysis

9.2.1 AntiNex AI Utilities

Standalone utilities for training AI. Used in:

https://github.com/jay-johnson/train-ai-with-django-swagger-jwt

Install

pip install antinex-utils

Development

1. Set up the repository

```
mkdir -p -m 777 /opt/antinex
git clone https://github.com/jay-johnson/antinex-utils.git /opt/antinex/utils
cd /opt/antinex/utils
```

2. Set up the virtual env and install

```
virtualenv -p python3 ~/.venvs/antinexutils && source ~/.venvs/antinexutils/bin/
↪activate && pip install -e .
```

Testing

Run all

```
python setup.py test
```

Run a test case

```
python -m unittest tests.test_classification.TestClassification.test_classification_
↪deep_dnn
```

```
python -m unittest tests.test_regression.TestRegression.test_dataset_regression_using_
↪scaler
```

AntiNex Stack Status

AntiNex AI Utilities is part of the AntiNex stack:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

Linting

flake8 .

pycodestyle –exclude=.tox,.eggs

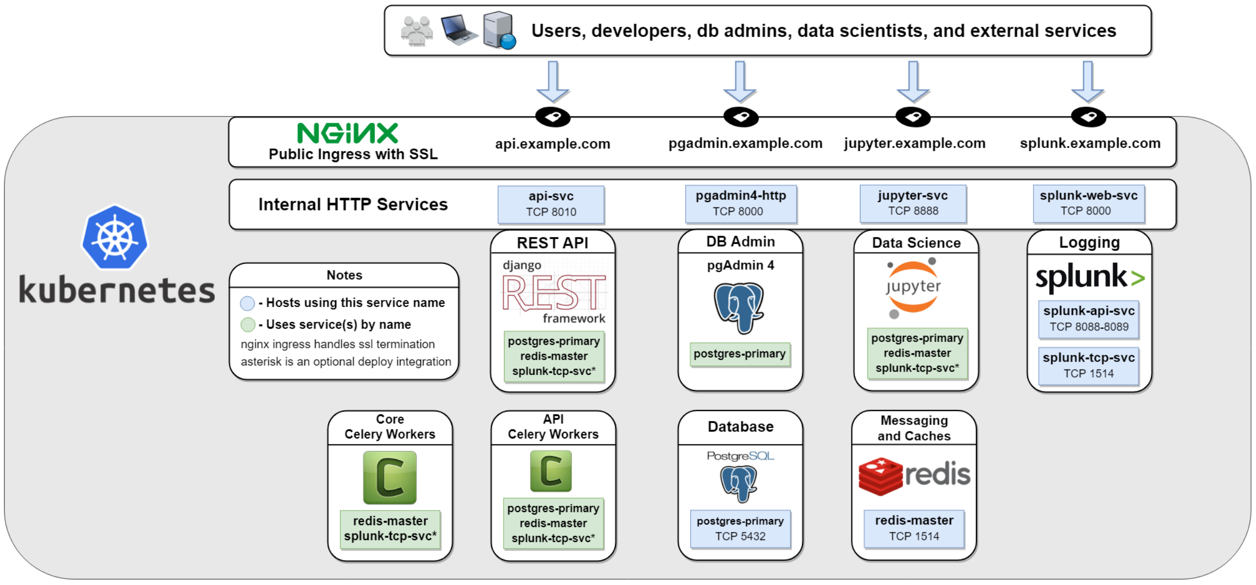
License

Apache 2.0 - Please refer to the LICENSE for more details

CHAPTER 10

API Reference

10.1 Deploying a Distributed AI Stack to Kubernetes on CentOS



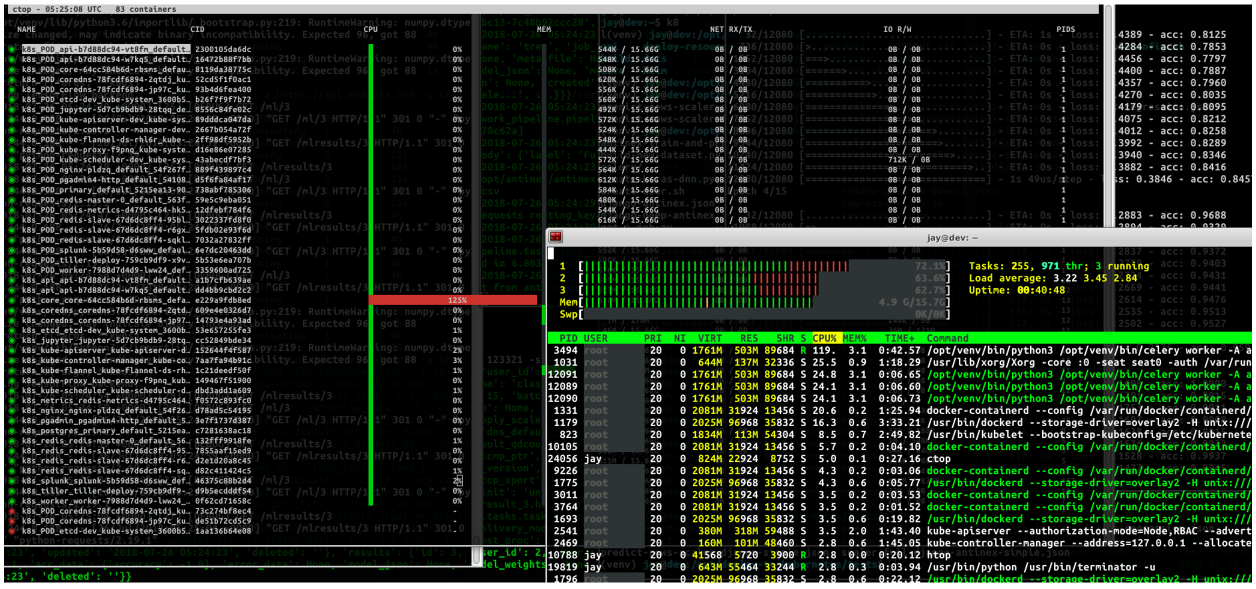
Install and manage a Kubernetes cluster with helm on a single CentOS 7 vm or in multi-host mode that runs the cluster on 3 CentOS 7 vms. Once running, you can deploy a distributed, scalable python stack capable of delivering a resilient REST service with JWT for authentication and Swagger for development. This service uses a decoupled REST API with two distinct worker backends for routing simple database read and write tasks vs long-running tasks that can use a Redis cache and do not need a persistent database connection. This is handy for not only simple CRUD applications and use cases, but also serving a secure multi-tenant environment where multiple users manage long-running tasks like training deep neural networks that are capable of making near-realtime predictions.

This guide was built for deploying the AntiNex stack of docker containers on a Kubernetes single host or multi-host cluster:

- Managing a Multi-Host Kubernetes Cluster with an External DNS Server
- Cert Manager with Let's Encrypt SSL support
- A Rook Ceph Cluster for Persistent Volumes
- Minio S3 Object Store
- Redis
- Postgres
- Django REST API with JWT and Swagger
- Django REST API Celery Workers
- Jupyter
- Core Celery Workers
- pgAdmin4
- (Optional) Splunk with TCP and HEC Service Endpoints

10.2 Getting Started

Note: Please ensure for single-vm hosting that the CentOS machine has at least 4 CPU cores and more than 8 GB ram. Here is a screenshot of the CPU utilization during AI training with only 3 cores:



```
ctop - 05:25:08 UTC   83 containers

 NAME                                     CID              CPU
 k8s_POD_api-b7d88dc94-vt8fm_default…     2300105da6dc      0%
 k8s_POD_api-b7d88dc94-w7kq5_default…     16472b88f7bb      0%
 k8s_POD_core-64cc584b6d-rbsms_defau…     8119da38775c      0%
 k8s_POD_coredns-78fcdf6894-2qtdj_ku…     52cd5f1f0ac1      0%
 k8s_POD_coredns-78fcdf6894-jp97c_ku…     93b4d6fea400      0%
 k8s_POD_etcd-dev_kube-system_3600b5…     b26f7f9f7b72      0%
 k8s_POD_jupyter-5d7cb9bdb9-28tqq_de…     8556c84fe02c      0%
 k8s_POD_kube-apiserver-dev_kube-sys…     89dddca047da      0%
 k8s_POD_kube-controller-manager-dev…     2667b054a72f      0%
 k8s_POD_kube-flannel-ds-rhl6r_kube-…     2ff98df5952b      0%
 k8s_POD_kube-proxy-f9pnq_kube-syste…     d16e86e07285      0%
 k8s_POD_kube-scheduler-dev_kube-sys…     43abecdf7bf3      0%
 k8s_POD_nginx-pldzq_default_54f267f…     889f439897c4      0%
 k8s_POD_pgadmin4-http_default_54108…     d5f6fa84af17      0%
 k8s_POD_primary_default_5215ea13-90…     738abf785306      0%
 k8s_POD_redis-master-0_default_563f…     59e5c9eba051      0%
 k8s_POD_redis-metrics-d4795c464-hk5…     12dfebf784f6      0%
 k8s_POD_redis-slave-67d6dc8ff4-95bl…     3022337fd8f0      0%
 k8s_POD_redis-slave-67d6dc8ff4-r6gx…     5fdb02e93f6d      0%
 k8s_POD_redis-slave-67d6dc8ff4-sqkl…     7032a27832ff      0%
 k8s_POD_splunk-5b59d58-d6sww_defaul…     6e7dc20463dd      0%
 k8s_POD_tiller-deploy-759cb9df9-x9v…     5b53e6ea707b      0%
 k8s_POD_worker-7988d7d4d9-lww24_def…     3359600ad725      0%
 k8s_api_api-b7d88dc94-vt8fm_default…     a1b7cfb639ae      0%
 k8s_api_api-b7d88dc94-w7kq5_default…     dd4bb9cbd2c2      0%
 k8s_core_core-64cc584b6d-rbsms_defa…     e229a9fdb8ed    125%
 k8s_coredns_coredns-78fcdf6894-2qtd…     609e4e0326d7      0%
 k8s_coredns_coredns-78fcdf6894-jp97…     14793e4a93ad      0%
 k8s_etcd_etcd-dev_kube-system_3600b…     53e657255fe3      1%
 k8s_jupyter_jupyter-5d7cb9bdb9-28tq…     cc52849bde34      0%
 k8s_kube-apiserver_kube-apiserver-d…     152644f4f587      2%
 k8s_kube-controller-manager_kube-co…     7aa7fa94b91c      3%
 k8s_kube-flannel_kube-flannel-ds-rh…     1c21deedf50f      1%
 k8s_kube-proxy_kube-proxy-f9pnq_kub…     149467f51900      0%
 k8s_kube-scheduler_kube-scheduler-d…     dbd3add1a609      1%
 k8s_metrics_redis-metrics-d4795c464…     f0572c893fc0      0%
 k8s_nginx_nginx-pldzq_default_54f26…     d78ad5c54195      0%
 k8s_pgadmin_pgadmin4-http_default_5…     3e7f1737d387      0%
 k8s_postgres_primary_default_5215ea…     c7281638ac18      0%
 k8s_redis_redis-master-0_default_56…     132fff9918fe      1%
 k8s_redis_redis-slave-67d6dc8ff4-95…     7855aaf15ed9      0%
 k8s_redis_redis-slave-67d6dc8ff4-r6…     d2e1d20a8c45      0%
 k8s_redis_redis-slave-67d6dc8ff4-sq…     d82c411424c5      1%
 k8s_splunk_splunk-5b59d58-d6sww_def…     46375c88b2d4      2%
 k8s_tiller_tiller-deploy-759cb9df9-…     d9b5ecdddf54      0%
 k8s_worker_worker-7988d7d4d9-lww24_…     0f62cd71658c      0%
 k8s_POD_coredns-78fcdf6894-2qtdj_ku…     73c274bf8ec4      -
 k8s_POD_coredns-78fcdf6894-jp97c_ku…     de51b72cd5c9      -
 k8s_POD_etcd-dev_kube-system_3600b5…     1aa136b64e08      -
```

```
4389 - acc: 0.8125
4284 - acc: 0.7853
4456 - acc: 0.7797
4400 - acc: 0.7887
4357 - acc: 0.7960
4270 - acc: 0.8035
4179 - acc: 0.8095
4075 - acc: 0.8212
4012 - acc: 0.8258
3992 - acc: 0.8289
3940 - acc: 0.8346
3882 - acc: 0.8416
s: 0.3846 - acc: 0.845

2883 - acc: 0.9688
```

```
jay@dev: ~

  1  [|||||||||||||||||||||||||||||||||||||||||||||||           72.1%]   Tasks: 255, 971 thr; 3 running
  2  [|||||||||||||||||||||||||||||||||||||||||                 63.6%]   Load average: 3.22 3.45 2.84
  3  [|||||||||||||||||||||||||||||||||||||||||                 62.7%]   Uptime: 00:40:48
  Mem[||||||||||||||||||||||||||||||||||||||              4.9 G/15.7G]
  Swp[                                                          0K/0K]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 3494 root       20   0 1761M  503M 89684 R 119.  3.1  0:42.57 /opt/venv/bin/python3 /opt/venv/bin/celery worker -A a
 1031 root       20   0  644M  137M 32336 S 25.5  0.9  1:18.29 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run
12091 root       20   0 1761M  503M 89684 S 24.8  3.1  0:06.65 /opt/venv/bin/python3 /opt/venv/bin/celery worker -A a
12089 root       20   0 1761M  503M 89684 S 24.1  3.1  0:06.60 /opt/venv/bin/python3 /opt/venv/bin/celery worker -A a
12090 root       20   0 1761M  503M 89684 S 24.1  3.1  0:06.73 /opt/venv/bin/python3 /opt/venv/bin/celery worker -A a
 1331 root       20   0 2081M 31924 13456 S 20.6  0.2  1:25.94 docker-containerd --config /var/run/docker/containerd/
 1179 root       20   0 2025M 96968 35832 S 16.3  0.6  3:33.21 /usr/bin/dockerd --storage-driver=overlay2 -H unix:///
  823 root       20   0 1834M  113M 54304 S  8.5  0.7  2:49.82 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernete
10105 root       20   0 2081M 31924 13456 S  5.7  0.2  0:04.10 docker-containerd --config /var/run/docker/containerd/
24056 jay        20   0  824M 22924  8752 S  5.0  0.1  0:27.16 ctop
 9226 root       20   0 2081M 31924 13456 S  4.3  0.2  0:03.06 docker-containerd --config /var/run/docker/containerd/
 1775 root       20   0 2025M 96968 35832 S  4.3  0.6  0:05.77 /usr/bin/dockerd --storage-driver=overlay2 -H unix:///
 3011 root       20   0 2081M 31924 13456 S  3.5  0.2  0:03.53 docker-containerd --config /var/run/docker/containerd/
 3764 root       20   0 2081M 31924 13456 S  3.5  0.2  0:01.52 docker-containerd --config /var/run/docker/containerd/
 1693 root       20   0 2025M 96968 35832 S  3.5  0.6  0:19.82 /usr/bin/dockerd --storage-driver=overlay2 -H unix:///
 2541 root       20   0  380M  318M 59488 S  3.5  2.0  1:43.40 kube-apiserver --authorization-mode=Node,RBAC --advert
 2469 root       20   0  160M  101M 48460 S  2.8  0.6  1:45.05 kube-controller-manager --address=127.0.0.1 --allocate
10788 jay        20   0 41568  5720  3900 R  2.8  0.0  0:20.12 htop
19819 jay        20   0  643M 55464 33244 R  2.8  0.3  0:03.94 /usr/bin/python /usr/bin/terminator -u
 1796 root       20   0 2025M 96968 35832 S  2.8  0.6  0:22.12 /usr/bin/dockerd --storage-driver=overlay2 -H unix:///
```

10.2.1 Overview

This guide installs the following systems and a storage solution Rook with Ceph cluster (default) or NFS volumes to prepare the host for running containers and automatically running them on host startup:

- Kubernetes
- Helm and Tiller

- Minio S3 Storage
- Persistent Storage Volumes using Rook with Ceph cluster or optional NFS Volumes mounted at: `/data/k8/redis`, `/data/k8/postgres`, `/data/k8/pgadmin`
- Flannel CNI

10.2.2 Install

Here is a video showing how to prepare the host to run a local Kubernetes cluster:

Preparing the host to run Kubernetes requires run this as root

```
sudo su
./prepare.sh
```

Note: This has only been tested on CentOS 7 and Ubuntu 18.04 and requires commenting out all swap entries in `/etc/fstab` to work

Warning: This guide used to install the cluster on Ubuntu 18.04, but after seeing high CPU utilization after a few days of operation this guide was moved to CentOS 7. The specific issues on Ubuntu were logged in `journalctl -xe` and appeared to be related to "volumes not being found" and "networking disconnects".

10.3 Validate

1. Install Kubernetes Config

   Run as your user

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   ```

   Or use the script:

   ```
   ./user-install-kubeconfig.sh
   ```

2. Check the Kubernetes Version

   ```
   kubectl version
   Client Version: version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.1",␣
   ↪GitCommit:"b1b29978270dc22fecc592ac55d903350454310a", GitTreeState:"clean",␣
   ↪BuildDate:"2018-07-17T18:53:20Z", GoVersion:"go1.10.3", Compiler:"gc", Platform:
   ↪"linux/amd64"}
   The connection to the server localhost:8080 was refused - did you specify the␣
   ↪right host or port?
   ```

3. Confirm the Kubernetes Pods Are Running

   ```
   kubectl get pods -n kube-system
   ```

```
NAME                              READY     STATUS    RESTARTS   AGE
coredns-78fcdf6894-k8srv          1/1       Running   0          4m
coredns-78fcdf6894-xx8bt          1/1       Running   0          4m
etcd-dev                          1/1       Running   0          3m
kube-apiserver-dev                1/1       Running   0          3m
kube-controller-manager-dev       1/1       Running   0          3m
kube-flannel-ds-m8k9w             1/1       Running   0          4m
kube-proxy-p4blg                  1/1       Running   0          4m
kube-scheduler-dev                1/1       Running   0          3m
tiller-deploy-759cb9df9-wxvp8     1/1       Running   0          4m
```

10.4 Deploy Redis and Postgres and the Nginx Ingress

Here is a video showing how to deploy Postgres, Redis, Nginx Ingress, and the pgAdmin4 as pods in the cluster:

Note: Postgres, pgAdmin4 and Redis use Rook Ceph to persist data

Here are the commands to deploy Postgres, Redis, Nginx Ingress, and pgAdmin4 in the cluster:

Note: Please ensure helm is installed and the tiller pod in the `kube-system` namespace is the `Running` state or Redis will encounter deployment issues

Install Go using the ./tools/install-go.sh script or with the commands:

```
# note go install has only been tested on CentOS 7 and Ubuntu 18.04:
sudo su
GO_VERSION="1.11"
GO_OS="linux"
GO_ARCH="amd64"
go_file="go${GO_VERSION}.${GO_OS}-${GO_ARCH}.tar.gz"
curl https://dl.google.com/go/${go_file} --output /tmp/${go_file}
export GOPATH=$HOME/go/bin
export PATH=$PATH:$GOPATH:$GOPATH/bin
tar -C $HOME -xzf /tmp/${go_file}
$GOPATH/go get github.com/blang/expenv
# make sure to add GOPATH and PATH to ~/.bashrc
```

```
./user-install-kubeconfig.sh
./deploy-resources.sh
```

If you want to deploy splunk you can add it as an argument:

```
./deploy-resources.sh splunk
```

If you want to deploy splunk with Let's Encrypt make sure to add `prod` as an argument:

```
./deploy-resources.sh splunk prod
```

10.5 Start Applications

Here is a video showing how to start the Django REST Framework, Celery Workers, Jupyter, and the AntiNex Core as pods in the cluster:

Start all applications as your user with the command:

```
./start.sh
```

If you want to deploy the splunk-ready application builds, you can add it as an argument:

```
./start.sh splunk
```

If you want to deploy the splunk-ready application builds integrated with Let's Encrypt TLS encryption, just add `prod` as an argument:

```
./start.sh splunk prod
```

Note: The Cert Manager is set to staging mode by default and requires the `prod` argument to prevent accidentally getting blocked due to Lets Encrypt rate limits

10.5.1 Confirm Pods are Running

Depending on how fast your network connection is the initial container downloads can take a few minutes. Please wait until all pods are `Running` before continuing.

```
kubectl get pods
```

10.6 Run a Database Migration

Here is a video showing how to apply database schema migrations in the cluster:

To apply new Django database migrations, run the following command:

```
./api/migrate-db.sh
```

10.7 Add Ingress Locations to /etc/hosts

When running locally (also known in these docs as `dev` mode), all ingress urls need to resolve on the network. Please append the following entries to your local `/etc/hosts` file on the `127.0.0.1` line:

```
sudo vi /etc/hosts
```

Append the entries to the existing `127.0.0.1` line:

```
127.0.0.1   <leave-original-values-here> api.example.com jupyter.example.com pgadmin.
↪example.com splunk.example.com s3.example.com ceph.example.com minio.example.com
```

10.8 Using the Minio S3 Object Store

By default, the Kubernetes cluster has a Minio S3 object store running on a Ceph Persistent Volume. S3 is a great solution for distributing files, datasets, configurations, static assets, build artifacts and many more across components, regions, and datacenters using an S3 distributed backend. Minio can also replicate some of the AWS Lambda event-based workflows with Minio bucket event listeners.

For reference, Minio was deployed using this script:

```
./minio/run.sh
```

10.8.1 View the Verification Tests on the Minio Dashboard

Login with:

- access key: `trexaccesskey`
- secret key: `trex123321`

https://minio.example.com/minio/s3-verification-tests/

10.8.2 Test Minio S3 with Bucket Creation and File Upload and Download

1. Run from inside the API container

   ```
   ./api/ssh.sh
   source /opt/venv/bin/activate && run_s3_test.py
   ```

   Example logs:

   ```
   creating test file: run-s3-test.txt
   connecting: http://minio-service:9000
   checking bucket=s3-verification-tests exists
   upload_file(run-s3-test.txt, s3-verification-tests, s3-worked-on-2018-08-12-15-21-
   ↪02)
   upload_file(s3-verification-tests, s3-worked-on-2018-08-12-15-21-02, download-run-
   ↪s3-test.txt)
   download_filename=download-run-s3-test.txt contents: tested on: 2018-08-12␣
   ↪15:21:02
   exit
   ```

2. Run from outside the Kubernetes cluster

   Note: This tool requires the python `boto3` pip is installed

   ```
   source ./minio/envs/ext.env
   ./minio/run_s3_test.py
   ```

3. Verify the files were uploaded to Minio

   https://minio.example.com/minio/s3-verification-tests/

10.9 Using the Rook Ceph Cluster

By default, the Kubernetes cluster is running a Rook Ceph cluster for storage which provides HA persistent volumes and claims.

You can review the persistent volumes and claims using the Ceph Dashboard:

https://ceph.example.com

10.10 Create a User

Create the user `trex` with password `123321` on the REST API.

```
./api/create-user.sh
```

10.11 Deployed Web Applications

Here are the hosted web application urls. These urls are made accessible by the included nginx-ingress.

10.12 View Django REST Framework

Login with:

- user: `trex`
- password: `123321`

https://api.example.com

10.13 View Swagger

Login with:

- user: `trex`
- password: `123321`

https://api.example.com/swagger

10.14 View Jupyter

Login with:

- password: `admin`

https://jupyter.example.com

10.15 View pgAdmin

Login with:

- user: `admin@admin.com`
- password: `123321`

https://pgadmin.example.com

10.16 View Minio S3 Object Storage

Login with:

- access key: `trexaccesskey`
- secret key: `trex123321`

https://minio.example.com

10.17 View Ceph

https://ceph.example.com

10.18 View Splunk

Login with:

- user: `trex`
- password: `123321`

https://splunk.example.com

10.19 Training AI with the Django REST API

These steps install the AntiNex python client for training a deep neural network to predict attack packets from recorded network data (all of which is already included in the docker containers).

1. Create a virtual environment and install the client

```
virtualenv -p python3 /opt/venv && source /opt/venv/bin/activate
pip install antinex-client
```

2. Watch the application logs

   From a separate terminal, you can tail the Django REST API logs with the command:

```
./api/logs.sh
```

   From a separate terminal, you can tail the Django Celery Worker logs with the command:

```
./worker/logs.sh
```

From a separate terminal, you can tail the AntiNex Core Worker logs with the command:

```
./core/logs.sh
```

Note: Use `ctrl + c` to stop these log tailing commands

10.20 Train a Deep Neural Network on Kubernetes

With virtual environment set up, we can use the client to train a deep neural network with the included datasets:

Note: this can take a few minutes to finish depending on your hosting resources

```
ai -a https://api.example.com -u trex -p 123321 -s -f ./tests/scaler-full-django-
↪antinex-simple.json
```

While you wait, here is a video showing the training and get results:

10.21 Get the AI Job Record

```
ai_get_job.py -a https://api.example.com -u trex -p 123321 -i 1
```

10.22 Get the AI Training Job Results

```
ai_get_results.py -a https://api.example.com -u trex -p 123321 -i 1 -s
```

10.23 Standalone Deployments

Below are steps to manually deploy each component in the stack with Kubernetes.

10.24 Deploy Redis

```
./redis/run.sh
```

Or manually with the commands:

```
echo "deploying persistent volume for redis"
kubectl apply -f ./redis/pv.yml
echo "deploying Bitnami redis stable with helm"
helm install \
```

(continues on next page)

(continued from previous page)

```
    --name redis stable/redis \
    --set rbac.create=true \
    --values ./redis/redis.yml
```

10.24.1 Confirm Connectivity

The following commands assume you have `redis-tools` installed (`sudo apt-get install redis-tools`).

```
redis-cli -h $(kubectl describe pod redis-master-0 | grep IP | awk '{print $NF}') -p␣
↪6379
10.244.0.81:6379> info
10.244.0.81:6379> exit
```

10.24.2 Debug Redis Cluster

1. Examine Redis Master

```
kubectl describe pod redis-master-0
```

2. Examine Persistent Volume Claim

```
kubectl get pvc
NAME                      STATUS    VOLUME                                     ␣
↪CAPACITY   ACCESS MODES   STORAGECLASS      AGE
redis-ceph-data           Bound     pvc-1a88e3a6-9df8-11e8-8047-0800270864a8   ␣
↪8Gi        RWO            rook-ceph-block   46m
```

3. Examine Persistent Volume

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM␣
↪POLICY   STATUS    CLAIM                             STORAGECLASS      REASON   ␣
↪ AGE
pvc-1a88e3a6-9df8-11e8-8047-0800270864a8   8Gi        RWO            Delete       ␣
↪     Bound     default/redis-ceph-data           rook-ceph-block             46m
```

10.24.3 Possible Errors

1. Create the Persistent Volumes

```
Warning  FailedMount       2m                 kubelet, dev       MountVolume.SetUp␣
↪failed for volume "redis-pv" : mount failed: exit status 32
```

```
./pvs/create-pvs.sh
```

10.24.4 Delete Redis

```
helm del --purge redis
release "redis" deleted
```

10.24.5 Delete Persistent Volume and Claim

1. Delete Claim

   ```
   kubectl delete pvc redis-data-redis-master-0
   ```

2. Delete Volume

   ```
   kubectl delete pv redis-pv
   persistentvolume "redis-pv" deleted
   ```

10.25 Deploy Postgres

10.25.1 Install Go

Using Crunchy Data's postgres containers requires having go installed. Go can be installed using the ./tools/install-go.sh script or with the commands:

```
# note go install has only been tested on CentOS 7 and Ubuntu 18.04:
sudo su
GO_VERSION="1.11"
GO_OS="linux"
GO_ARCH="amd64"
go_file="go${GO_VERSION}.${GO_OS}-${GO_ARCH}.tar.gz"
curl https://dl.google.com/go/${go_file} --output /tmp/${go_file}
export GOPATH=$HOME/go/bin
export PATH=$PATH:$GOPATH:$GOPATH/bin
tar -C $HOME -xzf /tmp/${go_file}
$GOPATH/go get github.com/blang/expenv
# make sure to add GOPATH and PATH to ~/.bashrc
```

10.25.2 Start

Start the Postgres container within Kubernetes:

```
./postgres/run.sh
```

10.25.3 Debug Postgres

1. Examine Postgres

   ```
   kubectl describe pod primary

   Type     Reason      Age    From                  Message
   ----     -------     ----   ----                  -------
   Normal   Scheduled   2m     default-scheduler     Successfully assigned default/primary␣
    ↪to dev
   ```

(continues on next page)

(continued from previous page)

```
Normal  Pulling    2m    kubelet, dev       pulling image "crunchydata/crunchy-
↪postgres:centos7-10.4-1.8.3"
Normal  Pulled     2m    kubelet, dev       Successfully pulled image
↪"crunchydata/crunchy-postgres:centos7-10.4-1.8.3"
Normal  Created    2m    kubelet, dev       Created container
Normal  Started    2m    kubelet, dev       Started container
```

2. Examine Persistent Volume Claim

```
kubectl get pvc
NAME                      STATUS    VOLUME                                     ␣
↪CAPACITY   ACCESS MODES   STORAGECLASS      AGE
pgadmin4-http-data        Bound     pvc-19031825-9df8-11e8-8047-0800270864a8   ␣
↪400M       RWX            rook-ceph-block   46m
primary-pgdata            Bound     pvc-17652595-9df8-11e8-8047-0800270864a8   ␣
↪400M       RWX            rook-ceph-block   46m
```

3. Examine Persistent Volume

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM␣
↪POLICY   STATUS    CLAIM                          STORAGECLASS      REASON   ␣
↪ AGE
pvc-17652595-9df8-11e8-8047-0800270864a8   400M       RWX            Delete       ␣
↪     Bound     default/primary-pgdata          rook-ceph-block             47m
pvc-19031825-9df8-11e8-8047-0800270864a8   400M       RWX            Delete       ␣
↪     Bound     default/pgadmin4-http-data      rook-ceph-block             47m
```

10.26 Deploy pgAdmin

Please confirm go is installed with the Install Go section.

10.26.1 Start

Start the pgAdmin4 container within Kubernetes:

```
./pgadmin/run.sh
```

10.26.2 Get Logs

```
./pgadmin/logs.sh
```

10.26.3 SSH into pgAdmin

```
./pgadmin/ssh.sh
```

10.27 Deploy Django REST API

Use these commands to manage the Django REST Framework pods within Kubernetes.

10.27.1 Start

```
./api/run.sh
```

10.27.2 Run a Database Migration

To apply a django database migration run the following command:

```
./api/migrate-db.sh
```

10.27.3 Get Logs

```
./api/logs.sh
```

10.27.4 SSH into the API

```
./api/ssh.sh
```

10.28 Deploy Django Celery Workers

Use these commands to manage the Django Celery Worker pods within Kubernetes.

10.28.1 Start

```
./worker/run.sh
```

10.28.2 Get Logs

```
./worker/logs.sh
```

10.28.3 SSH into the Worker

```
./worker/ssh.sh
```

10.29 Deploy AntiNex Core

Use these commands to manage the Backend AntiNex Core pods within Kubernetes.

10.29.1 Start

```
./core/run.sh
```

10.29.2 Get Logs

```
./core/logs.sh
```

10.29.3 SSH into the API

```
./core/ssh.sh
```

10.30 Deploy Jupyter

Use these commands to manage the Jupyter pods within Kubernetes.

10.30.1 Start

```
./jupyter/run.sh
```

10.30.2 Login to Jupyter

Login with:

- password: `admin`

https://jupyter.example.com

10.30.3 Get Logs

```
./jupyter/logs.sh
```

10.30.4 SSH into Jupyter

```
./jupyter/ssh.sh
```

10.31 Deploy Splunk

Use these commands to manage the Splunk container within Kubernetes.

10.31.1 Start

```
./splunk/run.sh
```

10.31.2 Login to Splunk

Login with:

- user: `trex`
- password: `123321`

https://splunk.example.com

10.32 Searching in Splunk

Here is the splunk searching command line tool I use with these included applications:

https://github.com/jay-johnson/spylunking

With search example documentation:

https://spylunking.readthedocs.io/en/latest/scripts.html#examples

10.33 Search using Spylunking

Find logs in splunk using the `sp` command line tool:

```
sp -q 'index="antinex" | reverse' -u trex -p 123321 -a $(./splunk/get-api-fqdn.sh) -i␣
↪antinex
```

10.34 Find Django REST API Logs in Splunk

```
sp -q 'index="antinex" AND name=api | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

10.35 Find Django Celery Worker Logs in Splunk

```
sp -q 'index="antinex" AND name=worker | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

10.36 Find Core Logs in Splunk

```
sp -q 'index="antinex" AND name=core | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

10.37 Find Jupyter Logs in Splunk

```
sp -q 'index="antinex" AND name=jupyter | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

Example for debugging `sp` splunk connectivity from inside an API Pod:

```
kubectl exec -it api-59496ccb5f-2wp5t -n default echo 'starting search' && /bin/bash -
↪c "source /opt/venv/bin/activate && sp -q 'index="antinex" AND hostname=local' -u␣
↪trex -p 123321 -a 10.101.107.205:8089 -i antinex"
```

10.37.1 Get Logs

```
./splunk/logs.sh
```

10.37.2 SSH into Splunk

```
./splunk/ssh.sh
```

10.38 Deploy Nginx Ingress

This project is currently using the nginx-ingress instead of the Kubernetes Ingress using nginx. Use these commands to manage and debug the nginx ingress within Kubernetes.

Note: The default Yaml file annotations only work with the nginx-ingress customizations

10.38.1 Start

```
./ingress/run.sh
```

10.38.2 Get Logs

```
./ingress/logs.sh
```

10.38.3 SSH into the Ingress

```
./ingress/ssh.sh
```

10.39 View Ingress Nginx Config

When troubleshooting the nginx ingress, it is helpful to view the nginx configs inside the container. Here is how to view the configs:

```
./ingress/view-configs.sh
```

10.40 View a Specific Ingress Configuration

If you know the pod name and the namespace for the nginx-ingress, then you can view the configs from the command line with:

```
app_name="jupyter"
app_name="pgadmin"
app_name="api"
use_namespace="default"
pod_name=$(kubectl get pods -n ${use_namespace} | awk '{print $1}' | grep nginx |␣
↪head -1)
kubectl exec -it ${pod_name} -n ${use_namespace} cat /etc/nginx/conf.d/${use_
↪namespace}-${app_name}-ingress.conf
```

10.41 Deploy Splunk

10.41.1 Start

To deploy splunk you can add the argument `splunk` to the ./deploy-resources.sh splunk script. Or you can manually run it with the command:

```
./splunk/run.sh
```

Or if you want to use Let's Encrypt for SSL:

```
./splunk/run.sh prod
```

10.42 Deploy Splunk-Ready Applications

After deploying the splunk pod, you can deploy the splunk-ready applications with the command:

```
./start.sh splunk
```

10.42.1 Get Logs

```
./splunk/logs.sh
```

10.42.2 SSH into Splunk

```
./splunk/ssh.sh
```

10.42.3 View Ingress Config

```
./splunk/view-ingress-config.sh
```

10.43 Create your own self-signed x509 TLS Keys, Certs and Certificate Authority with Ansible

If you have openssl installed you can use this ansible playbook to create your own certificate authority (CA), keys and certs.

1. Create the CA, Keys and Certificates

   ```
   cd ansible
   ansible-playbook -i inventory_dev create-x509s.yml
   ```

2. Check the CA, x509, keys and certificates for the client and server were created

   ```
   ls -l ./ssl
   ```

10.44 Deploying Your Own x509 TLS Encryption files as Kubernetes Secrets

This is a work in progress, but in `dev` mode the cert-manager is not in use. Instead the cluster utilizes pre-generated x509s TLS SSL files created with the included ansible playbook create-x509s.yml. Once created, you can deploy them as Kubernetes secrets using the deploy-secrets.sh script and reload them at any time in the future.

10.44.1 Deploy Secrets

Run this to create the TLS secrets:

```
./ansible/deploy-secrets.sh
```

10.44.2 List Secrets

```
kubectl get secrets | grep tls
tls-ceph                kubernetes.io/tls                     2         36m
tls-client              kubernetes.io/tls                     2         36m
tls-database            kubernetes.io/tls                     2         36m
tls-docker              kubernetes.io/tls                     2         36m
tls-jenkins             kubernetes.io/tls                     2         36m
tls-jupyter             kubernetes.io/tls                     2         36m
tls-k8                  kubernetes.io/tls                     2         36m
tls-kafka               kubernetes.io/tls                     2         36m
tls-kibana              kubernetes.io/tls                     2         36m
tls-minio               kubernetes.io/tls                     2         36m
tls-nginx               kubernetes.io/tls                     2         36m
tls-pgadmin             kubernetes.io/tls                     2         36m
tls-phpmyadmin          kubernetes.io/tls                     2         36m
tls-rabbitmq            kubernetes.io/tls                     2         36m
tls-redis               kubernetes.io/tls                     2         36m
tls-restapi             kubernetes.io/tls                     2         36m
tls-s3                  kubernetes.io/tls                     2         36m
tls-splunk              kubernetes.io/tls                     2         36m
tls-webserver           kubernetes.io/tls                     2         36m
```

10.44.3 Reload Secrets

If you want to deploy new TLS secrets at any time, use the `reload` argument with the `deploy-secrets.sh` script. Doing so will delete the original secrets and recreate all of them using the new TLS values:

```
./ansible/deploy-secrets.sh -r
```

10.45 Deploy Cert Manager with Let's Encrypt

Use these commands to manage the Cert Manager with Let's Encrypt SSL support within Kubernetes. By default, the cert manager is deployed only in `prod` mode. If you run it in production mode, then it will install real, valid x509 certificates from Let's Encrypt into the nginx-ingress automatically.

10.45.1 Start with Let's Encrypt x509 SSL Certificates

Start the cert manager in `prod` mode to enable Let's Encrypt TLS Encryption with the command:

```
./start.sh prod
```

Or manually with the command:

```
./cert-manager/run.sh prod
```

If you have splunk you can just add it to the arguments:

```
./start.sh splunk prod
```

10.45.2 View Logs

When using the production mode, make sure to view the logs to ensure you are not being blocked due to rate limiting:

```
./cert-manager/logs.sh
```

10.46 Stop the Cert Manager

If you notice things are not working correctly, you can quickly prevent yourself from getting blocked by stopping the cert manager with the command:

```
./cert-manager/_uninstall.sh
```

Note: If you get blocked due to rate-limits it will show up in the cert-manager logs like:

```
I0731 07:53:43.313709       1 sync.go:273] Error issuing certificate for default/api.
↪antinex.com-tls: error getting certificate from acme server: acme:␣
↪urn:ietf:params:acme:error:rateLimited: Error finalizing order :: too many␣
↪certificates already issued for exact set of domains: api.antinex.com: see https://
↪letsencrypt.org/docs/rate-limits/
E0731 07:53:43.313738       1 sync.go:182] [default/api.antinex.com-tls] Error␣
↪getting certificate 'api.antinex.com-tls': secret "api.antinex.com-tls" not found
```

10.46.1 Debugging

To reduce debugging issues, the cert manager ClusterIssuer objects use the same name for staging and production mode. This is nice because you do not have to update all the annotations to deploy on production vs staging:

The cert manager starts and defines the issuer name for both production and staging as:

```
--set ingressShim.defaultIssuerName=letsencrypt-issuer
```

Make sure to set any nginx ingress annotations that need Let's Encrypt SSL encryption to these values:

```
annotations:
  kubernetes.io/tls-acme: "true"
  kubernetes.io/ingress.class: "nginx"
  certmanager.k8s.io/cluster-issuer: "letsencrypt-issuer"
```

10.47 Troubleshooting

10.48 Customize Minio and How to Troubleshoot

10.48.1 Change the Minio Access and Secret Keys

1. Change the secrets file: `minio/secrets/default_access_keys.yml`

   Change the `access_key` and `secret_key` values after generating the new base64 string values for the secrets file:

```
echo -n "NewAccessKey" | base64
TmV3QWNjZXNzS2V5
# now you can replace the access_key's value in the secrets file with the string:␣
↪TmV3QWNjZXNzS2V5
```

```
echo -n "NewSecretKey" | base64
TmV3U2VjcmV0S2V5
# now you can replace the secret_key's value in the secrets file with the string:␣
↪TmV3QWNjZXNzS2V5
```

2. Deploy the secrets file

```
kubectl apply -f ./minio/secrets/default_access_keys.yml
```

3. Restart the Minio Pod

```
kubectl delete pod -l app=minio
```

If you have changed the default access and secret keys, then you will need to export the following environment variables as needed to make sure the `./minio/run_s3_test.py` test script works:

```
export S3_ACCESS_KEY=<minio access key: trexaccesskey - default>
export S3_SECRET_KEY=<minio secret key: trex123321 - default>
export S3_REGION_NAME=<minio region name: us-east-1 - default>
export S3_ADDRESS=<minio service endpoint: external address found with the script ./
↪minio/get-s3-endpoint.sh and the internal cluster uses the service: minio-
↪service:9000>
# examples of setting up a minio env files are in: ./minio/envs
```

10.48.2 View the Minio Dashboard

Login with:

- access key: `trexaccesskey`
- secret key: `trex123321`

https://minio.example.com

10.48.3 Get S3 Internal Endpoint

If you want to use the Minio S3 service within the cluster please use the endpoint:

```
minio-service:9000
```

or source the internal environment file:

```
source ./minio/envs/int.env
```

10.48.4 Get S3 External Endpoint

If you want to use the Minio S3 service from outside the cluser please use the endpoint provided by the script:

```
./minio/get-s3-endpoint.sh
# which for this documentation was the minio service's Endpoints:
# 10.244.0.103:9000
```

or source the external environment file:

```
source ./minio/envs/ext.env
```

10.48.5 Debugging Steps

1. Load the Minio S3 external environment variables:

   ```
   source ./minio/envs/ext.env
   ```

2. Run the S3 Verification test script

   ```
   ./minio/run_s3_test.py
   ```

3. Confirm Verification Keys are showing up in this Minio S3 bucket

   https://minio.example.com/minio/s3-verification-tests/

   If not please use the describe tools in `./minio/describe-*.sh` to grab the logs and please file a GitHub issue

10.48.6 Describe Pod

```
./minio/describe-service.sh
```

10.48.7 Describe Service

```
./minio/describe-service.sh
```

10.48.8 Describe Ingress

```
./minio/describe-ingress.sh
```

10.48.9 Uninstall Minio

```
./minio/_uninstall.sh
```

10.49 Ceph Troubeshooting

Please refer to the Rook Common Issues for the latest updates on how to use your Rook Ceph cluster.

Note: By default Ceph is not hosting the S3 solution unless `cephs3` is passed in as an argument to `deploy-resource.sh`.

There are included troubleshooting tools in the `./rook` directory with an overview of each below:

10.49.1 Validate Ceph System Pods are Running

```
./rook/view-system-pods.sh

-----------------------------------------
Getting the Rook Ceph System Pods:
kubectl -n rook-ceph-system get pod
NAME                                  READY     STATUS    RESTARTS   AGE
rook-ceph-agent-g9vzm                 1/1       Running   0          7m
rook-ceph-operator-78d498c68c-tbsdf   1/1       Running   0          7m
rook-discover-h9wj9                   1/1       Running   0          7m
```

10.49.2 Validate Ceph Pods are Running

```
./rook/view-ceph-pods.sh

-----------------------------------------
Getting the Rook Ceph Pods:
kubectl -n rook-ceph get pod
NAME                                  READY     STATUS      RESTARTS   AGE
rook-ceph-mgr-a-9c44495df-7jksz       1/1       Running     0          6m
rook-ceph-mon0-rxxsl                  1/1       Running     0          6m
rook-ceph-mon1-gqblg                  1/1       Running     0          6m
rook-ceph-mon2-7xfsq                  1/1       Running     0          6m
rook-ceph-osd-id-0-7d4d4c8794-kgr2d   1/1       Running     0          6m
rook-ceph-osd-prepare-dev-kmsn9       0/1       Completed   0          6m
rook-ceph-tools                       1/1       Running     0          6m
```

10.49.3 Validate Persistent Volumes are Bound

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   ␣
↪STATUS    CLAIM                              STORAGECLASS      REASON    AGE
pvc-03e6e4ef-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete           ␣
↪Bound     default/certs-pv-claim             rook-ceph-block             46m
pvc-0415de24-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete           ␣
↪Bound     default/configs-pv-claim           rook-ceph-block             46m
pvc-0441307f-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete           ␣
↪Bound     default/datascience-pv-claim       rook-ceph-block             46m
pvc-0468ef73-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete           ␣
↪Bound     default/frontendshared-pv-claim    rook-ceph-block             46m
pvc-04888222-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete           ␣
↪Bound     default/staticfiles-pv-claim       rook-ceph-block             46m
pvc-1c3e359d-9df8-11e8-8047-0800270864a8   10Gi       RWO            Delete           ␣
↪Bound     default/minio-pv-claim             rook-ceph-block             46m
```

10.49.4 Validate Persistent Volume Claims are Bound

```
kubectl get pvc
NAME                      STATUS    VOLUME                                     ␣
↪CAPACITY   ACCESS MODES   STORAGECLASS      AGE
certs-pv-claim            Bound     pvc-03e6e4ef-9df8-11e8-8047-0800270864a8   1Gi        ␣
↪    RWO            rook-ceph-block   47m
configs-pv-claim          Bound     pvc-0415de24-9df8-11e8-8047-0800270864a8   1Gi        ␣
↪    RWO            rook-ceph-block   47m
datascience-pv-claim      Bound     pvc-0441307f-9df8-11e8-8047-0800270864a8   1Gi        ␣
↪    RWO            rook-ceph-block   47m
frontendshared-pv-claim   Bound     pvc-0468ef73-9df8-11e8-8047-0800270864a8   1Gi        ␣
↪    RWO            rook-ceph-block   47m
minio-pv-claim            Bound     pvc-1c3e359d-9df8-11e8-8047-0800270864a8   10Gi       ␣
↪    RWO            rook-ceph-block   46m
```

10.49.5 Create a Persistent Volume Claim

Going forward, Ceph will automatically create a persistent volume if one is not available for binding to an available Persistent Volume Claim. To create a new persistent volume, just create a claim and verify the Rook Ceph cluster created the persistent volume and both are bound to each other.

```
kubectl apply -f pvs/pv-staticfiles-ceph.yml
```

10.49.6 Verify the Persistent Volume is Bound

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   ␣
↪STATUS    CLAIM                            STORAGECLASS      REASON    AGE
pvc-77afbc7a-9ade-11e8-b293-0800270864a8   20Gi       RWO            Delete           ␣
↪Bound     default/staticfiles-pv-claim    rook-ceph-block             2s
```

10.49.7 Verify the Persistent Volume Claim is Bound

```
kubectl get pvc
NAME                   STATUS    VOLUME                                     CAPACITY  ␣
↪ ACCESS MODES   STORAGECLASS      AGE
staticfiles-pv-claim   Bound     pvc-77afbc7a-9ade-11e8-b293-0800270864a8   20Gi      ␣
↪ RWO            rook-ceph-block   11s
```

10.49.8 Describe Persistent Volumes

```
kubectl describe pv pvc-c88fc37b-9adf-11e8-9fae-0800270864a8
Name:            pvc-c88fc37b-9adf-11e8-9fae-0800270864a8
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by=ceph.rook.io/block
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    rook-ceph-block
Status:          Bound
```

(continues on next page)

(continued from previous page)

```
Claim:            default/certs-pv-claim
Reclaim Policy:   Delete
Access Modes:     RWO
Capacity:         20Gi
Node Affinity:    <none>
Message:
Source:
    Type:       FlexVolume (a generic volume resource that is provisioned/attached␣
↪using an exec based plugin)
    Driver:     ceph.rook.io/rook-ceph-system
    FSType:     xfs
    SecretRef:  <nil>
    ReadOnly:   false
    Options:    map[clusterNamespace:rook-ceph image:pvc-c88fc37b-9adf-11e8-9fae-
↪0800270864a8 pool:replicapool storageClass:rook-ceph-block]
Events:         <none>
```

10.49.9 Show Ceph Cluster Status

```
./rook/show-ceph-status.sh

----------------------------------------------
Getting the Rook Ceph Status with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools ceph status
cluster:
    id:     7de1988c-03ea-41f3-9930-0bde39540552
    health: HEALTH_OK

services:
    mon: 3 daemons, quorum rook-ceph-mon2,rook-ceph-mon0,rook-ceph-mon1
    mgr: a(active)
    osd: 1 osds: 1 up, 1 in

data:
    pools:   1 pools, 100 pgs
    objects: 12 objects, 99 bytes
    usage:   35443 MB used, 54756 MB / 90199 MB avail
    pgs:     100 active+clean
```

10.49.10 Show Ceph OSD Status

```
./rook/show-ceph-osd-status.sh

----------------------------------------------
Getting the Rook Ceph OSD Status with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools ceph osd status
+----+-------------------------------------+-------+-------+--------+---------+-------
↪-+---------+-----------+
| id |                 host                |  used | avail | wr ops | wr data | rd␣
↪ops | rd data |   state   |
+----+-------------------------------------+-------+-------+--------+---------+-------
↪-+---------+-----------+
| 0  | rook-ceph-osd-id-0-7d4d4c8794-kgr2d | 34.6G | 53.4G |    0   |     0   |    0 ␣
↪ |      0    | exists,up |
```

(continues on next page)

(continued from previous page)

```
+----+--------------------------------------+--------+--------+---------+----------+--------
↪-+----------+------------+
```

10.49.11 Show Ceph Free Space

```
./rook/show-ceph-df.sh

-----------------------------------------------
Getting the Rook Ceph df with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools ceph df
GLOBAL:
    SIZE        AVAIL       RAW USED     %RAW USED
    90199M      54756M       35443M         39.29
POOLS:
    NAME            ID      USED      %USED     MAX AVAIL     OBJECTS
    replicapool     1         99         0         50246M          12
```

10.49.12 Show Ceph RDOS Free Space

```
./rook/show-ceph-rados-df.sh

-----------------------------------------------
Getting the Rook Ceph rados df with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools rados df
POOL_NAME   USED OBJECTS CLONES COPIES MISSING_ON_PRIMARY UNFOUND DEGRADED RD_OPS RD ␣
↪ WR_OPS WR
replicapool   99      12       0      12                  0        0         0    484␣
↪381k     17 7168

total_objects    12
total_used       35443M
total_avail      54756M
total_space      90199M
```

10.49.13 Out of IP Addresses

Flannel can exhaust all available ip addresses in the CIDR network range. When this happens please run the following command to clean up the local cni network files:

```
./tools/reset-flannel-cni-networks.sh
```

10.50 AntiNex Stack Status

Here are the AntiNex repositories, documentation and build reports:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

10.51 Reset Cluster

Here is a video showing how to reset the local Kubernetes cluster.

Please be careful as these commands will shutdown all containers and reset the Kubernetes cluster.

Run as root:

```
sudo su
kubeadm reset -f
./prepare.sh
```

Or use the file:

```
sudo su
./tools/cluster-reset.sh
```

Or the full reset and deploy once ready:

```
sudo su
cert_env=dev; ./tools/reset-flannel-cni-networks.sh; ./tools/cluster-reset.sh ; ./
↪user-install-kubeconfig.sh ; sleep 30; ./deploy-resources.sh splunk ${cert_env}
exit
# as your user
./user-install-kubeconfig.sh
# depending on testing vs prod:
# ./start.sh splunk
# ./start.sh splunk prod
```

10.52 Development

Right now, the python virtual environment is only used to bring in ansible for running playbooks, but it will be used in the future with the kubernetes python client as I start using it more and more.

```
virtualenv -p python3 /opt/venv && source /opt/venv/bin/activate && pip install -e .
```

10.53 Testing

```
py.test
```

or

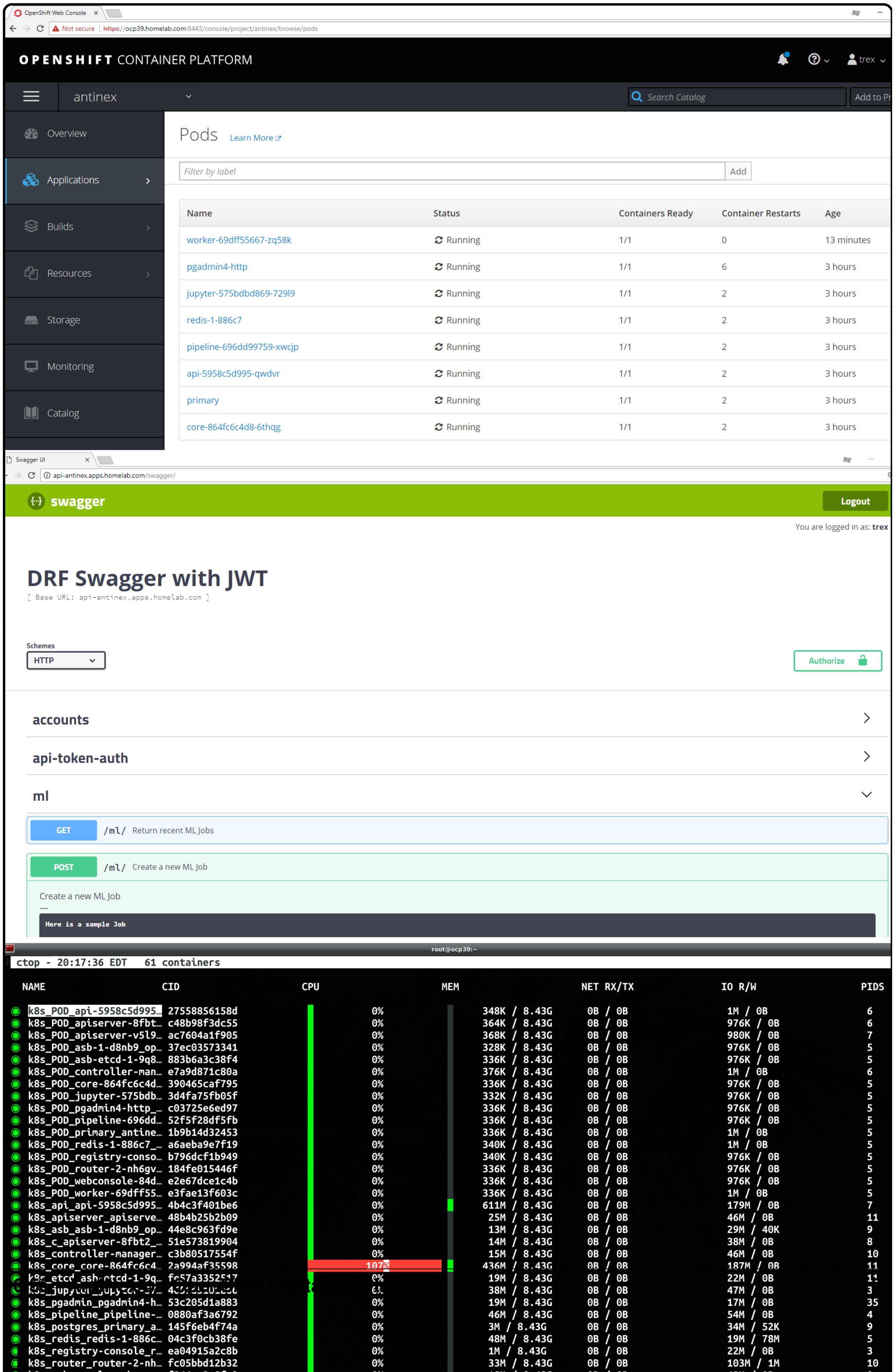
```
python setup.py test
```

10.54 License

Apache 2.0 - Please refer to the LICENSE for more details

10.55 AntiNex on OpenShift Container Platform

Here is a guide for running the AntiNex stack on OpenShift Container Platform. This was tested on version 3.9.



Pods

Name	Status	Containers Ready	Container Restarts	Age
worker-69dff55667-zq58k	Running	1/1	0	13 minutes
pgadmin4-http	Running	1/1	6	3 hours
jupyter-575bdbd869-729l9	Running	1/1	2	3 hours
redis-1-886c7	Running	1/1	2	3 hours
pipeline-696dd99759-xwcjp	Running	1/1	2	3 hours
api-5958c5d995-qwdvr	Running	1/1	2	3 hours
primary	Running	1/1	2	3 hours
core-864fc6c4d8-6thqg	Running	1/1	2	3 hours

You are logged in as: trex

DRF Swagger with JWT

[Base URL: api-antinex.apps.homelab.com]

GET /ml/ Return recent ML Jobs

POST /ml/ Create a new ML Job

Create a new ML Job

Here is a sample Job

```
ctop - 20:17:36 EDT   61 containers

 NAME                   CID             CPU          MEM               NET RX/TX     IO R/W        PIDS

 k8s_POD_api-5958c5d995… 27558856158d   0%    348K / 8.43G   0B / 0B   1M / 0B      6
 k8s_POD_apiserver-8fbt… c48b98f3dc55   0%    364K / 8.43G   0B / 0B   976K / 0B    6
 k8s_POD_apiserver-v5l9… ac7604a1f905   0%    368K / 8.43G   0B / 0B   980K / 0B    7
 k8s_POD_asb-1-d8nb9_op… 37ec03573341   0%    328K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_asb-etcd-1-9q8… 883b6a3c38f4   0%    336K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_controller-man… e7a9d871c80a   0%    376K / 8.43G   0B / 0B   1M / 0B      6
 k8s_POD_core-864fc6c4d… 390465caf795   0%    336K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_jupyter-575bdb… 3d4fa75fb05f   0%    332K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_pgadmin4-http_… c03725e6ed97   0%    336K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_pipeline-696dd… 52f5f28df5fb   0%    336K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_primary_antine… 1b9b14d32453   0%    336K / 8.43G   0B / 0B   1M / 0B      5
 k8s_POD_redis-1-886c7_… a6aeba9e7f19   0%    340K / 8.43G   0B / 0B   1M / 0B      5
 k8s_POD_registry-conso… b796dcf1b949   0%    340K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_router-2-nh6gv… 184fe015446f   0%    336K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_webconsole-84d… e2e67dce1c4b   0%    336K / 8.43G   0B / 0B   976K / 0B    5
 k8s_POD_worker-69dff55… e3fae13f603c   0%    336K / 8.43G   0B / 0B   1M / 0B      5
 k8s_api_api-5958c5d995… 4b4c3f401be6   0%    611M / 8.43G   0B / 0B   179M / 0B    7
 k8s_apiserver_apiserve… 48b4b25b2b09   0%     25M / 8.43G   0B / 0B   46M / 0B     11
 k8s_asb_asb-1-d8nb9_op… 44e8c963fd9e   0%     13M / 8.43G   0B / 0B   29M / 40K    9
 k8s_c_apiserver-8fbt2_… 51e573819904   0%     14M / 8.43G   0B / 0B   38M / 0B     8
 k8s_controller-manager… c3b80517554f   0%     15M / 8.43G   0B / 0B   46M / 0B     10
 k8s_core_core-864fc6c4… 2a994af35598   107%  436M / 8.43G   0B / 0B   187M / 0B    11
 k8s_etcd_asb-etcd-1-9q… fe57a3352517   0%     19M / 8.43G   0B / 0B   22M / 0B     11
 k8s_jupyter_jupyter-57… [illegible]    0%     38M / 8.43G   0B / 0B   47M / 0B     3
 k8s_pgadmin_pgadmin4-h… 53c205d1a883   0%     19M / 8.43G   0B / 0B   17M / 0B     35
 k8s_pipeline_pipeline-… 0880af3a6792   0%     46M / 8.43G   0B / 0B   54M / 0B     4
 k8s_postgres_primary_a… 145f6eb4f74a   0%      3M / 8.43G   0B / 0B   34M / 52K    9
 k8s_redis_redis-1-886c… 04c3f0cb38fe   0%     48M / 8.43G   0B / 0B   19M / 78M    5
 k8s_registry-console_r… ea04915a2c8b   0%      1M / 8.43G   0B / 0B   22M / 0B     3
 k8s_router_router-2-nh… fc05bbd12b32   0%     33M / 8.43G   0B / 0B   103M / 1M    10
```

This will deploy the following containers to OpenShift Container Platform:

1. API Server - Django REST Framework with JWT and Swagger
2. API Workers - Celery Workers to support the Django REST API
3. Core Worker - AntiNex AI Core Celery Worker
4. Jupyter - Includes ready-to-use AntiNex IPython Notebooks
5. Pipeline - AntiNex Network Pipeline Celery Worker
6. Posgres 10.4 - Crunchy Data Single Primary
7. Redis 3.2
8. pgAdmin4

10.55.1 Getting Started

1. Clone

   This guide assumes the repository is cloned to the directory:

   /opt/antinex/api

   ```
   mkdir -p -m 777 /opt/antinex
   git clone https://github.com/jay-johnson/train-ai-with-django-swagger-jwt.git /
   ↪opt/antinex/api
   ```

2. Setting up Database Tools

   For preparing Ubuntu 18 to manage the Crunchy containers:

   ```
   sudo apt install golang-go
   mkdir -p -m 777 /opt/antinex
   # on ubuntu 18.04:
   export GOPATH=$HOME/go
   export PATH=$PATH:$GOROOT/bin:$GOPATH/bin
   go get github.com/blang/expenv
   ```

3. Enable Admin Rights for Users

   On the OpenShift Container Platform, add `cluster-admin` role to all users that need to deploy AntiNex on OCP

   ```
   [root@ocp39 ~]# oc adm policy add-cluster-role-to-user cluster-admin trex
   cluster role "cluster-admin" added: "trex"
   [root@ocp39 ~]#
   ```

4. Persistent Volumes

   For Postgres and Redis to use a persistent volume, the user must be a **cluster-admin**.

5. Resources

   Please make sure to give the hosting vm(s) enough memory to run the stack. If you are using OpenShift Container Platform please use at least 2 CPU cores and 8 GB of RAM.

6. Set up /etc/hosts

   OpenShift Container Platform is running on a vm with an ip: **192.168.0.35** and with these application fqdns in `/etc/hosts`.

```
192.168.0.35    ocp39.homelab.com api-antinex.apps.homelab.com jupyter-antinex.
↪apps.homelab.com postgres-antinex.apps.homelab.com redis-antinex.apps.homelab.
↪com primary-antinex.apps.homelab.com pgadmin4-http-antinex.apps.homelab.com
```

10.55.2 Login to OpenShift Container Platform

Here's an example of logging into OpenShift Container Platform

```
oc login https://ocp39.homelab.com:8443
```

10.55.3 Deploy

Deploy the containers to OpenShift Container Platform

Run Deployment Command

```
./deploy.sh
```

If you have Splunk set up you can deploy the splunk deployment configs with the command:

```
./deploy.sh splunkenterprise
```

10.55.4 Check the AntiNex Stack

You can view the **antinex** project's pod on the OpenShift web console:

OpenShift Container Platform:

https://ocp39.homelab.com:8443/console/project/antinex/browse/pods

You can also use the command line:

```
oc status -v
```

```
oc status -v
In project antinex on server https://ocp39.homelab.com:8443

http://api-antinex.apps.homelab.com to pod port 8010 (svc/api)
deployment/api deploys jayjohnson/antinex-api:latest
    deployment #1 running for 7 hours - 1 pod

http://jupyter-antinex.apps.homelab.com to pod port 8888 (svc/jupyter)
deployment/jupyter deploys jayjohnson/antinex-jupyter:latest
    deployment #1 running for 7 hours - 1 pod

http://pgadmin4-http-antinex.apps.homelab.com to pod port pgadmin4-http (svc/pgadmin4-
↪http)
pod/pgadmin4-http runs crunchydata/crunchy-pgadmin4:centos7-10.3-1.8.2

http://primary-antinex.apps.homelab.com to pod port 5432 (svc/primary)
pod/primary runs crunchydata/crunchy-postgres:centos7-10.4-1.8.3

http://redis-antinex.apps.homelab.com to pod port 6379-tcp (svc/redis)
```

(continues on next page)

(continued from previous page)

```
dc/redis deploys istag/redis:latest
    deployment #1 deployed 7 hours ago - 1 pod

deployment/core deploys jayjohnson/antinex-core:latest
deployment #1 running for 7 hours - 1 pod

deployment/pipeline deploys jayjohnson/antinex-pipeline:latest
deployment #1 running for 7 hours - 1 pod

deployment/worker deploys jayjohnson/antinex-worker:latest
deployment #1 running for 7 hours - 1 pod

Info:
* pod/pgadmin4-http has no liveness probe to verify pods are still running.
    try: oc set probe pod/pgadmin4-http --liveness ...
* pod/primary has no liveness probe to verify pods are still running.
    try: oc set probe pod/primary --liveness ...
* deployment/api has no liveness probe to verify pods are still running.
    try: oc set probe deployment/api --liveness ...
* deployment/core has no liveness probe to verify pods are still running.
    try: oc set probe deployment/core --liveness ...
* deployment/jupyter has no liveness probe to verify pods are still running.
    try: oc set probe deployment/jupyter --liveness ...
* deployment/pipeline has no liveness probe to verify pods are still running.
    try: oc set probe deployment/pipeline --liveness ...
* deployment/worker has no liveness probe to verify pods are still running.
    try: oc set probe deployment/worker --liveness ...
* dc/redis has no readiness probe to verify pods are ready to accept traffic or␣
↪ensure deployment is successful.
    try: oc set probe dc/redis --readiness ...
* dc/redis has no liveness probe to verify pods are still running.
    try: oc set probe dc/redis --liveness ...

View details with 'oc describe <resource>/<name>' or list everything with 'oc get all
↪'.
```

10.55.5 Migrations

Migrations have to run inside an **api** container. Below is a recording of running the initial migration.

OpenShift Container Platform

The command from the video is included in the openshift directory, and you can run the command to show how to run a migration. Once the command finishes, you can copy and paste the output into your shell to quickly run a migration:

```
./show-migrate-cmds.sh

Run a migration with:
oc rsh api-5958c5d995-jjxkt
/bin/bash
. /opt/venv/bin/activate && cd /opt/antinex/api && source /opt/antinex/api/envs/
↪openshift-no-hostnames.env && export POSTGRES_HOST=primary && export POSTGRES_
↪DB=webapp && export POSTGRES_USER=antinex && export POSTGRES_PASSWORD=antinex && ./
↪run-migrations.sh
exit
exit
```

10.55.6 Creating a User

Here's how to create the default user **trex**

OpenShift Container Platform

1. Create a User from the command line

   The commands to create the default user **trex** are:

   ```
   source users/user_1.sh
   ./create-user.sh
   ```

2. Create a User using Swagger

   You can create users using swagger the API's swagger url (here's the default one during creation of this guide):

   http://api-antinex.apps.homelab.com/swagger/

3. Create a User from a User file

   You can create your own user file's like: **users/user_1.sh** that have the supported environment keys in a file before running. You can also just exported them in the current shell session (but having a resource file will be required in the future):

   Here's the steps to build your own:

   (a) Find the API Service

   ```
   $ oc status | grep svc/api
   http://api-antinex.apps.homelab.com to pod port 8010 (svc/api)
   ```

   (b) Confirm it is Discovered by the AntiNex Get API URL Tool

   ```
   $ /opt/antinex/api/openshift/get-api-url.sh
   http://api-antinex.apps.homelab.com
   ```

   (c) Set the Account Details

   ```
   export API_USER="trex"
   export API_PASSWORD="123321"
   export API_EMAIL="bugs@antinex.com"
   export API_FIRSTNAME="Guest"
   export API_LASTNAME="Guest"
   export API_URL=http://api-antinex.apps.homelab.com
   export API_VERBOSE="true"
   export API_DEBUG="false"
   ```

   (d) Create the user

   ```
   ./create-user.sh <optional path to user file>
   ```

   (e) Get a JWT Token for the New User

   ```
   ./get-token.sh
   ```

Train a Deep Neural Network

Here's how to train a deep neural network using the AntiNex Client and the Django AntiNex dataset:

10.55.7 Commands for Training a Deep Neural Network on OpenShift with AntiNex

1. Install the AntiNex Client

   ```
   pip install antinex-client
   ```

2. Source User File

   ```
   source ./users/user_1.sh
   ```

3. Train the Deep Neural Network with the Django Dataset

   ```
   ai_train_dnn.py -f ../tests/scaler-full-django-antinex-simple.json -s
   ```

4. Get the Job

   The job from the video was MLJob.id: 3

   ```
   ai_get_job.py -i 3
   ```

5. Get the Job Result

   The job's result from the video was MLJobResult.id: 3

   ```
   ai_get_results.py -i 3
   ```

10.55.8 Drop and Restore Database with the Latest Migration

You can drop the database and restore it to the latest migration with this command. Copy and paste the output to run the commands quickly. Make sure to get the second batch or using the `./show-migrate-cmds.sh` if you need to migrate at some point in the future.

```
./tools/drop-database.sh
```

10.55.9 Debugging

Here is how to debug AntiNex on OpenShift. This is a work in progress so please feel free to reach out if you see a problem that is not documented here.

10.55.10 Drill Down into the Splunk Logs

If you deployed AntiNex with Splunk, then can use the Spylunking - sp command line tool or use the Splunk web app: http://splunkenterprise:8000/en-US/app/search/search

10.55.11 Find API Logs in Splunk

Find the API's logs by using the deployment config environment variables with the command:

```
sp -q 'index="antinex" AND name="api" | head 5 | reverse'
creating client user=trex address=splunkenterprise:8089
connecting trex@splunkenterprise:8089
2018-06-26 22:07:08,971 ml-sz - INFO - MLJob get user_id=2 pk=4
```

(continues on next page)

(continued from previous page)

```
2018-06-26 22:07:08,976 ml-sz - INFO - MLJob get res={'status': 0, 'code': 200, 'er
2018-06-26 22:07:09,011 ml - INFO - mljob_result get
2018-06-26 22:07:09,012 ml-sz - INFO - MLJobResults get user_id=2 pk=4
2018-06-26 22:07:11,458 ml-sz - INFO - MLJobResults get res={'status': 0, 'code': 200,
↪ 'er
done
```

10.55.12 Find Worker Logs in Splunk

Find the Worker's logs by using the deployment config environment variables with the command:

```
sp -q 'index="antinex" AND name="worker" | head 5 | reverse'
creating client user=trex address=splunkenterprise:8089
connecting trex@splunkenterprise:8089
2018-06-26 22:07:01,990 ml_prc_results - INFO - APIRES updating job_id=4 result_id=4
2018-06-26 22:07:01,991 ml_prc_results - INFO - saving job_id=4
2018-06-26 22:07:02,003 ml_prc_results - INFO - saving result_id=4
2018-06-26 22:07:07,898 ml_prc_results - INFO - APIRES done
2018-06-26 22:07:07,899 celery.app.trace - INFO - Task drf_network_pipeline.pipeline.
↪tasks.task_ml_process_results[5499207f-4faa-430e-89ec-c136829da902] succeeded in 6.
↪908605030999752s: None
done
```

10.55.13 Find Core Logs in Splunk

Find the Core's logs by using the deployment config environment variables with the command:

```
sp -q 'index="antinex" AND name="core" | head 5 | reverse'
creating client user=trex address=splunkenterprise:8089
connecting trex@splunkenterprise:8089
2018-06-26 22:06:55,834 send_results - INFO - sending response queue=drf_network_
↪pipeline.pipeline.tasks.task_ml_process_results task=drf_network_pipeline.pipeline.
↪tasks.task_ml_process_results retries=100000
2018-06-26 22:06:57,530 send_results - INFO - task.id=5499207f-4faa-430e-89ec-
↪c136829da902
2018-06-26 22:06:57,530 send_results - INFO - send_results_to_broker - done
2018-06-26 22:06:57,530 processor - INFO - CORERES Full-Django-AntiNex-Simple-Scaler-
↪DNN publishing results success=True
2018-06-26 22:06:57,531 processor - INFO - Full-Django-AntiNex-Simple-Scaler-DNN -␣
↪model=full-django-antinex-simple-scaler-dnn finished processing
done
```

10.55.14 Find Core AI Utilities Logs in Splunk

Find the Core's AntiNex Utility logs with the command:

```
sp -q 'index="antinex" AND name="core" AND make_predict | head 5 | reverse'
creating client user=trex address=splunkenterprise:8089
connecting trex@splunkenterprise:8089
2018-06-26 22:06:42,236 make_predict - INFO - Full-Django-AntiNex-Simple-Scaler-DNN -␣
↪ml_type=classification scores=[0.00016556291390729116, 0.9982615894039735]␣
↪accuracy=99.82615894039735 merging samples=30200 with predictions=30200 labels={'-1
↪': 'not_attack', '0': 'not_attack', '1': 'attack'}
```

(continues on next page)

(continued from previous page)

```
2018-06-26 22:06:48,017 make_predict - INFO - Full-Django-AntiNex-Simple-Scaler-DNN -␣
↪packaging classification predictions=30200 rows=30200
2018-06-26 22:06:48,017 make_predict - INFO - Full-Django-AntiNex-Simple-Scaler-DNN -␣
↪no image_file
2018-06-26 22:06:48,017 make_predict - INFO - Full-Django-AntiNex-Simple-Scaler-DNN -␣
↪created image_file=None
2018-06-26 22:06:48,018 make_predict - INFO - Full-Django-AntiNex-Simple-Scaler-DNN -␣
↪predictions done
done
```

10.55.15 Find Worker AI Utilities Logs in Splunk

Find the Worker's AntiNex Utility logs with the command:

```
sp -q 'index="antinex" AND name="worker" AND make_predict | head 5 | reverse'
creating client user=trex address=splunkenterprise:8089
connecting trex@splunkenterprise:8089
2018-06-26 21:45:04,351 make_predict - INFO - job_3_result_3 - merge_df=1651
2018-06-26 21:45:04,351 make_predict - INFO - job_3_result_3 - packaging regression␣
↪predictions=1651 rows=18
2018-06-26 21:45:04,352 make_predict - INFO - job_3_result_3 - no image_file
2018-06-26 21:45:04,352 make_predict - INFO - job_3_result_3 - created image_file=None
2018-06-26 21:45:04,352 make_predict - INFO - job_3_result_3 - predictions done
done
```

10.55.16 Tail API Logs

```
oc logs -f deployment/api
```

or

```
./logs-api.sh
```

10.55.17 Tail Worker Logs

```
oc logs -f deployment/worker
```

or

```
./logs-worker.sh
```

10.55.18 Tail AI Core Logs

```
oc logs -f deployment/core
```

or

```
./logs-core.sh
```

10.55.19 Tail Pipeline Logs

```
oc logs -f deployment/pipeline
```

or

```
./logs-pipeline.sh
```

10.55.20 Change the Entrypoint

To keep the containers running just add something like: `tail -f <some file>` to keep the container running for debugging issues.

I use:

```
&& tail -f /var/log/antinex/api/api.log
```

10.55.21 SSH into API Container

```
oc rsh deployment/api /bin/bash
```

10.55.22 SSH into API Worker Container

```
./ssh-worker.sh
```

or

```
oc rsh deployment/worker /bin/bash
```

10.55.23 SSH into AI Core Container

```
oc rsh deployment/core /bin/bash
```

10.55.24 Stop All Containers

Stop all the containers without changing the persistent volumes with the command:

```
./stop-all.sh
```

10.55.25 Delete Everything

Remove, delete and clean up everything in the AntiNex project with the command:

```
./remove-all.sh
```

Troubleshooting

10.55.26 Permission Errors for Postgres or Redis

If you see an error about permission denied in the logs for the primary postgres server or redis that mentions one of these directories:

```
/pgdata
/exports/redis-antinex
```

Then run this command to ssh over to the OCP vm and fix the volume mount directories. Please note, this tool assumes you have copied over the ssh keys and are using NFS mounts for OCP volumes.

```
./tools/delete-and-fix-volumes.sh
```

10.56 Source Code - ML Pipeline

These are the methods for developing with the current ML Pipeline app within the Django Rest Framework.

10.56.1 Constants

Constants for the ML

10.56.2 Building a Response Dictionary

This builds a dictionary that is published to the AntiNex Core within the MLJob's prediction manifest. This dictionary contains how to send the results back to the core. This would allow for an environment to run many Rest APIs and reuse the same core workers.

`drf_network_pipeline.pipeline.build_worker_result_node.`**`build_worker_result_node`**`(`*req=None*`)`

> **Parameters** **`req`** – incoming request dictionary - not used right now

10.56.3 Creating ML Job Stub Records for Tracking Purposes

Creates initial `MLJob` and `MLJobResult` record stub in the database

`drf_network_pipeline.pipeline.create_ml_job_record.`**`create_ml_job_record`**`(`*req_data=None*`)`

> **Parameters** **`req_data`** – dictionary to build the MLJob and MLJobResult objects

10.56.4 Creating New Training Datasets

Creates an initial `MLPrepare` record stub in the database

`drf_network_pipeline.pipeline.create_ml_prepare_record.`**`create_ml_prepare_record`**`(`*req_data=None*

> **Parameters** **`req_data`** – dictionary to build the MLPrepare object

10.56.5 Process AntiNex Core Worker Results

Fills in the `MLJob` and `MLJobResult` records with the JSON response from the AntiNex Core.

`drf_network_pipeline.pipeline.process_worker_results.`**`handle_worker_results_message`**`(`*body=Non*

Parameters **`body`** – contents from the results

`drf_network_pipeline.pipeline.process_worker_results.`**`process_worker_results`**`(`*res_node=None*`)`

Parameters **`res_node`** – incoming request dictionary - not used right now

10.56.6 Celery Tasks

Celery tasks that are handled within the Django Rest API Worker when the environment variable `CELERY_ENABLED` is set to `1`

`(task)``drf_network_pipeline.pipeline.tasks.`**`task_ml_job`**`(`*req_node=None*`)`

Parameters

- **`self`** – parent task object for bind=True
- **`req_node`** – job utils dictionary for passing a dictionary

`(task)``drf_network_pipeline.pipeline.tasks.`**`task_ml_prepare`**`(`*req_node=None*`)`

Parameters

- **`self`** – parent task object for bind=True
- **`req_node`** – job utils dictionary for passing a dictionary

`(task)``drf_network_pipeline.pipeline.tasks.`**`task_ml_process_results`**`(`*res_node=None*`)`

Core workers send results back to the REST API worker here

Parameters

- **`self`** – parent task object for bind=True
- **`res_node`** – results dictionary from the core

`(task)``drf_network_pipeline.pipeline.tasks.`**`task_publish_to_core`**`(`*publish_node=None*`)`

Parameters

- **`self`** – parent task object for bind=True
- **`publish_node`** – dictionary to send to the AntiNex Core Worker

10.56.7 Utility Methods

Utility methods

`drf_network_pipeline.pipeline.utils.`**`convert_to_date`**`(`*value=None, format='%Y-%m-%d %H:%M:%S'*`)`

param: value - datetime object param: format - string format

10.57 Source Code - Job Helpers

These are the helper methods for abstracting celery calls from the Django REST Framework Serializers. These are optional for most users, I just find them helpful because the serializers all examine a common dictionary structure instead of custom ones all over the code. The response structure is:

```
task_response_node = {
    "status": status,
    "err": err,
    "task_name": task_name,
    "data": data,
    "celery_enabled": celery_enabled,
    "use_cache": use_cache,
    "cache_key": cache_key
}
```

1. **status** will be a const value from the **drf_network_pipeline.pipeline.consts**

   Response Status Codes

   ```
   SUCCESS = 0
   FAILED = 1
   ERR = 2
   EX = 3
   NOTRUN = 4
   INVALID = 5
   NOTDONE = 6
   ```

2. **err** will be an empty string on **SUCCESS** and not-empty if there was a problem
3. **data** is the result from the Celery worker (if it was used instead of **python manage.py runserver 0.0.0.0:8010**)
4. **use_cache** is a flag meaning the results ere also cached in the **cache_key** for **django-cacheops** to use (this is not supported yet)
5. **task_name** is a human readable task label for debugging in the logs

10.57.1 Build Task Request

`drf_network_pipeline.job_utils.build_task_request.`**`build_task_request`**(*status=4, err='not-set', task_name='', data=None, job_id=None, celery_enabled=False, use_cache=False, cache_record=False, cache_key=None*)

build_task_node

Builds a common request dictionary for all Celery tasks being wrapped with the utils framework

Parameters

- **`status`** – task return status code

- **`err`** – task error message for debugging
- **`task_name`** – task label for debugging
- **`data`** – task return data
- **`job_id`** – task job id
- **`celery_enabled`** – control flag for testing celery tasks
- **`use_cache`** – use the cached record if available
- **`cache_record`** – cache the result in redis after done
- **`cache_key`** – cache the result in this redis key

10.57.2 Build Task Response

`drf_network_pipeline.job_utils.build_task_response.`**`build_task_response`**`(`*status=4, err='not-set', task_name='', data=None, celery_enabled=False, use_cache=False, cache_key=None*`)`

Builds a common response dictionary for all Celery tasks being wrapped with the utils framework

Parameters

- **`status`** – task return status code
- **`err`** – task error message for debugging
- **`task_name`** – task label for debugging
- **`data`** – task return data
- **`celery_enabled`** – control flag for testing celery tasks
- **`use_cache`** – use the cached record if available
- **`cache_key`** – cache the result in this redis key

10.57.3 Handle Task Method

`drf_network_pipeline.job_utils.handle_task_method.`**`handle_task_method`**`(`*req_node=None, task_method=None, get_result=False, delay_timeout=1.0*`)`

Wraps task invocation for easier debugging with a standardized dictionary status, error, data response

Parameters

- **`req_node`** – request tracking data
- **`task_method`** – task method to run
- **`get_result`** – get the result from task
- **`delay_timeout`** – timeout in seconds to wait

10.57.4 Run Task

`drf_network_pipeline.job_utils.run_task.`**`run_task`**(*task_method=None*, *task_name='please-set-name'*, *req_data=None*, *get_result=False*, *delay_timeout=1.0*, *use_cache=False*, *cache_record=False*, *cache_key=None*)

Handles Celery sync/async task processing

Parameters

- **`task_method`** – requested method
- **`task_name`** – name of the task for logging
- **`req_data`** – requested data
- **`get_result`** – get the result from task
- **`delay_timeout`** – seconds to wait for the task to finish
- **`use_cache`** – use the cached record if available
- **`cache_record`** – cache the result in redis after done
- **`cache_key`** – cache the result in this redis key

10.58 Source Code - Django Rest Framework Serializers

10.58.1 User Serializers

These are the current User Serializers

`class` `drf_network_pipeline.sz.user.`**`UserSerializer`**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

User Serializer

`create`(*request*, *validated_data*)

Parameters **`validated_data`** – post dict

`delete`(*request*, *pk*)

Delete a User

Parameters

- **`request`** – http request
- **`pk`** – User.id

`get`(*request*, *pk*)

Get user

Parameters

- **`request`** – http request
- **`pk`** – User.id

`lookup_user`(*user_id*)

Parameters **`user_id`** – user id

`update`(*request*, *validated_data*, *pk=None*)
Update User

Parameters

- **`request`** – http request
- **`validated_data`** – dict of values
- **`pk`** – User.id

10.58.2 ML Serializers

These are the current ML Serializers

class `drf_network_pipeline.sz.ml.`**`MLPrepareSerializer`**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)
AntiNex Prepare Dataset Serializer

`create`(*request*, *validated_data*)
Start a new Prepare Job

Parameters

- **`request`** – http request
- **`validated_data`** – post dictionary

`delete`(*request*, *pk*)
Delete an MLPrepare

Parameters

- **`request`** – http request
- **`pk`** – MLPrepare.id

`get`(*request*, *pk*)
Get MLPrepare record

Parameters

- **`request`** – http request
- **`pk`** – MLPrepare.id

`update`(*request*, *validated_data*, *pk=None*)
Update an MLPrepare

Parameters

- **`request`** – http request
- **`validated_data`** – dict of values
- **`pk`** – MLPrepare.id

class `drf_network_pipeline.sz.ml.`**`MLJobsSerializer`**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)
AntiNex AI Job Serializer

create (*request*, *validated_data*)
Start a new MLJob

Parameters

- **request** – http request
- **validated_data** – post dictionary

delete (*request*, *pk*)
Delete an MLJob

Parameters

- **request** – http request
- **pk** – MLJob.id

get (*request*, *pk*)
Get MLJob or Get Recent ML Jobs for User (if pk=None)

Parameters

- **request** – http request
- **pk** – MLJob.id

update (*request*, *validated_data*, *pk=None*)
Update an MLJob

Parameters

- **request** – http request
- **validated_data** – dict of values
- **pk** – MLJob.id

class drf_network_pipeline.sz.ml.**MLJobResultsSerializer** (*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

AntiNex AI Job Results Serializer

create (*request*, *validated_data*)
Create new MLJobResult

Parameters

- **request** – http request
- **validated_data** – post dictionary

delete (*request*, *pk*)
Delete an MLJobResult

Parameters

- **request** – http request
- **pk** – MLJobResult.id

get (*request*, *pk*)
Get MLResult record

Parameters

- **request** – http request

- **pk** – MLJobResult.id

update (*request*, *validated_data*, *pk=None*)
Update an MLJobResult

Parameters

- **request** – http request
- **validated_data** – dict of values
- **pk** – MLJobResult.id

10.59 Source Code - Database Models

10.59.1 AntiNex DB Models

Here are the `MLJob`, `MLJobResult` and `MLPrepare` classes.

class drf_network_pipeline.pipeline.models.**MLJob** (*id*, *user*, *title*, *desc*, *ds_name*, *algo_name*, *ml_type*, *status*, *control_state*, *predict_feature*, *predict_manifest*, *training_data*, *pre_proc*, *post_proc*, *meta_data*, *tracking_id*, *version*, *created*, *updated*, *deleted*)

exception DoesNotExist

exception MultipleObjectsReturned

class drf_network_pipeline.pipeline.models.**MLJobResult** (*id*, *user*, *job*, *status*, *test_size*, *csv_file*, *meta_file*, *acc_data*, *error_data*, *model_json*, *model_weights*, *model_weights_file*, *acc_image_file*, *predictions_json*, *version*, *created*, *updated*, *deleted*)

exception DoesNotExist

exception MultipleObjectsReturned

class drf_network_pipeline.pipeline.models.**MLPrepare** (*id*, *user*, *status*, *control_state*, *title*, *desc*, *full_file*, *clean_file*, *meta_suffix*, *output_dir*, *ds_dir*, *ds_glob_path*, *pipeline_files*, *post_proc*, *label_rules*, *meta_data*, *tracking_id*, *version*, *created*, *updated*, *deleted*)

exception DoesNotExist

exception MultipleObjectsReturned

10.59.2 User DB Model

class drf_network_pipeline.users.models.**User**(*id*, *password*, *last_login*, *is_superuser*, *username*, *first_name*, *last_name*, *email*, *is_staff*, *is_active*, *date_joined*)

exception DoesNotExist

exception MultipleObjectsReturned

10.60 Frequently Asked Questions

10.60.1 What AntiNex is Not and Disclaimers

There's a lot of moving pieces in AI, and I wanted to be clear what is currently not supported:

1. Custom layers or custom Deep Neural Network models - only Keras Sequential neural networks, KerasRegressor, KerasClassifier, Stratified Kfolds, cross validation scoring, Scalers, Add and Dropout are supported. PR's are always welcomed!
2. Able to tell what your applications are doing today that is good, non-attack traffic out of the box. AntiNex requires recording how the network is being used in normal operation + identifying what you want to protect (do you want tcp traffic only? or a combination of tcp + udp + arp?). It uses the captured traffic to build the intial training dataset.
3. Exotic attacks - The network pipeline includes the Zed Attack Proxy (ZED) for OWASP dynamic security analysis. This tool attacks using a fuzzing attack on web applications. ZED was used to generate the latest attack datasets, and there is no guarantee the latest dnn's will always be effective with attacks I have not seen yet. Please share your findings and reach out if you know how to generate new, better attack simulations to help us all. PR's are always welcomed!
4. Image predictions and Convoluted Neural Networks - it's only works on numeric datasets.
5. Recurrent Neural Networks - I plan on adding LTSM support into the antinex-utils, but the scores were already good enough to release this first build.
6. Embedding Layers - I want to add payload deserialization to the packet processing with support for decrypting traffic, but the dnn scores were good enough to skip this feature for now.
7. Adversarial Neural Networks - I plan on creating attack neural networks from the datasets to beat up the trained ones, but this is a 2.0 feature at this point.
8. Saving models to disk is broken - I have commented out the code and found a keras issue that looks like the same problem I am hitting… I hope it is resovled so we can share model files via S3.

10.60.2 Why the name?

I was describing what this did and my sister-in-law said it reminded her of antivirus but for network defense. So instead of calling it **Anti-Network Exploits** it's just **AntiNex** or **anex** for short. Thanks Alli for the name!

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

CHAPTER 12

What AntiNex is Not

There's a lot of moving pieces in AI, and I wanted to be clear what is currently not supported:

1. Custom layers or custom Deep Neural Network models - only Keras Sequential neural networks, KerasRegressor, KerasClassifier, Stratified Kfolds, cross validation scoring, Scalers, Add and Dropout are supported. PR's are always welcomed!
2. Able to tell what your applications are doing today that is good, non-attack traffic out of the box. AntiNex requires recording how the network is being used in normal operation + identifying what you want to protect (do you want tcp traffic only? or a combination of tcp + udp + arp?). It uses the captured traffic to build the initial training dataset.
3. Exotic attacks - The network pipeline includes the Zed Attack Proxy (ZAP) for OWASP dynamic security analysis. This tool attacks using a fuzzing attack on web applications. ZAP was used to generate the latest attack datasets, and there is no guarantee the latest dnn's will always be effective with attacks I have not seen yet. Please share your findings and reach out if you know how to generate new, better attack simulations to help us all. PR's are always welcomed!
4. Image predictions and Convoluted Neural Networks - it's only works on numeric datasets.
5. Recurrent Neural Networks - I plan on adding LTSM support into the antinex-utils, but the scores were already good enough to release this first build.
6. Embedding Layers - I want to add payload deserialization to the packet processing with support for decrypting traffic, but the dnn scores were good enough to skip this feature for now.
7. Adversarial Neural Networks - I plan on creating attack neural networks from the datasets to beat up the trained ones, but this is a 2.0 feature at this point.
8. Saving models to disk is broken - I have commented out the code and found a keras issue that looks like the same problem I am hitting… I hope it's fixed soon so we can share model files via S3.

CHAPTER 13

Disclaimers and Legal

1. This is a tool that requires capturing your network traffic to be effective. I am not legally responsible for any damaging or incriminating network traffic you record.
2. I am not legally responsible for where you deploy this tool. It is meant to help educate how to defend.
3. This is still an emerging technology, and I am not claiming it will work to defend everything out there on the internet. It works very well for predicting when an attack using OWASP fuzzing attacks are targeting web applications. I am not legally responsible if you run this and you still get hacked, lose data, lose your job, lose your money, destroyed personal property or anything worse. I built it to educate how to build your own deep neural networks to defend. It will forever be an ongoing battle and arms race with malicious actors on the internet trying to beat every claimed-unbeatable fortress.

Python Module Index

Index