

netAI Getting Started

DESCRIPTION

The Network Traffic based Application Identification (netAI) has been developed for identifying the end host applications that are responsible for traffic flows in the network. Unlike previous solutions that identify the application based on port numbers or packet payload (either through protocol decoding or signatures) netAI computes a variety of payload independent features (e.g. packet length statistics) for a traffic flow and uses machine learning (ML) techniques to identify the application that generated a particular traffic flow. ML is a discipline of the wider area of Artificial Intelligence (AI). Before netAI can be used to classify a particular application it must be trained on a representative set of traffic flows of that application. netAI can be used offline (reading packet data from tracefile) and online (live capturing on network interface). NetAI can be used on Linux and BSD operating systems.

This document is a getting started guide for netAI. As netAI is heavily based on two tools called Weka and NetMate we also provide some introduction to them. For installation instructions see the *netAI Installation Guide*.

HOME

The official homepage for netAI is <http://caia.swin.edu.au/urp/dstc/netai/>. Please check this page for any software updates, FAQ etc. before contacting the developers.

The netAI package can be downloaded here: <http://caia.swin.edu.au/urp/dstc/netai/download>

WEKA

Weka is a machine learning suite written in Java that contains implementations of a large number of machine learning algorithms. We use Weka to create classification models and classify data. Weka can be accessed using a GUI, command line or include as a library within Java software.

A really good primer for using Weka on the command line, and also creating java applications around Weka, is located at <http://alex.seewald.at/WEKA/> (February 2006).

The GUI version of Weka can be executed with the command:

```
java -jar weka.jar
```

netAI does not use the Weka GUI, although the command line syntax for netAI is similar to that used with Weka. Therefore is important to understand how to use Weka at the command line, and there are some nice examples on the aforementioned primer page to help. You might run Weka as follows:

```
java -Xmx1024M -classpath ./path/to/weka.jar weka.classifier.to.use -flags.
```

For example, run `java -Xmx1024M -classpath /path/to/weka.jar weka.classifiers.trees.J48` for a list of options for the J48 classifier. It is possible to add `weka.jar` to your java classpath so that you do not need to specify `-classpath` each time.

Datasets used in Weka are in the 'ARFF' file format, described at <http://www.cs.waikato.ac.nz/~ml/weka/arff.html> (February 2006). Example arff files can be obtained from the Weka homepage. We have included some example ARFF formatted files for testing with netAI, and it is important to note the relationship between the ARFF files and the statistics output from NetMate (and how to create ARFF files from NetMate output).

NETMATE

Typically NetMate is either used for live capturing on a network interface (e.g. eth0 on Linux) or reading data from a trace file (e.g. tcpdump). To tell NetMate what to do with incoming packet data a ruleset is needed. The netAI distribution contains an example ruleset in the etc subdirectory called netAI.xml. NetMate can then export statistics via TCP, UDP or write to file.

NetMate can be executed for capturing on a network interface directly:

```
superuser@machine: netmate -i eth0 -r netAI.xml
```

Live capturing requires running NetMate as superuser unless the network interface has been made accessible for ordinary users. Alternatively NetMate can read packet data from a tcpdump or Endance Record Format file:

```
superuser@machine: netmate -f somedumpfile -r netAI.xml
```

Both of these commands will make NetMate run in the current console. The NetMate distribution provides a wrapper to run NetMate in the background called nmrsh, for example the following command will start a live capture in the background:

```
superuser@machine: nmrsh start -i eth0 -r netAI.xml
```

This can be stopped by executing the following command:

```
superuser@machine: nmrsh stop
```

The manual explains configuration files and switches etc, and can be found at the NetMate homepage at: <http://www.ip-measurement.org/tools/netmate/index.php?p=documentation> (February 2006)

NETAI

The netAI script in the bin directory can be used to run netAI. It executes both NetMate and netAI specific code. It is also possible to directly run the netAI code for example:

```
Java -classpath .:<weka path>/weka.jar testClassifier weka.classifiers.trees.J48 -c 1 -t training-file.arff -bh
```

This will train a new J48 decision tree classifier based on the training file specified and then listen for incoming instance data.

STEP-BY-STEP EXAMPLES

Taking a look around

This example has been installed to the default installation directory `install`. The listing for the directory should look like this:

```
nwilliams@ccurp3:~/tools/netai/install> ll
drwxr-xr-x  2 nwilliams users    4096 2006-01-24 17:33 bin
drwxr-xr-x  4 nwilliams users    4096 2006-01-24 17:33 etc
drwxr-xr-x  3 nwilliams users    4096 2006-01-24 17:32 lib
drwxr-xr-x  3 nwilliams users    4096 2006-01-24 17:33 man
-rw-r--r--  1 nwilliams users 982822 2005-12-08 17:48 netmate-0.9.3.tar.gz
drwxr-xr-x  2 nwilliams users    4096 2006-01-24 17:33 share
drwxr-xr-x  3 nwilliams users    4096 2006-01-24 17:29 src
drwxr-xr-x  4 nwilliams users    4096 2006-01-24 17:32 var
drwxr-xr-x  5 nwilliams users    4096 2005-03-07 02:32 weka-3-4-4
-rw-r--r--  1 nwilliams users 8547671 2005-03-08 10:35 weka-3-4-4.zip
```

Some directories of interest are listed below:

- bin contains the various executables/scripts for running netAI and NetMate.

- etc contains configuration files for NetMate and netAI.
- man contains the netAI manpage

You shouldn't have to play with the configuration files for the moment, as it is likely you can accomplish what you want by changing the `netAI-rules.xml` file.

An example netAI rules file is shown below, and was used to create the example datasets from a tcpdump file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RULESET SYSTEM "rulefile.dtd">
<RULESET ID="1">
  <!-- global part is the default for all rules -->
  <!-- overwritten by rule specific configuration -->
  <GLOBAL>

    <ACTION NAME="netai_flowstats">
      <PREF NAME="Idle_Threshold">1000000</PREF>
    </ACTION>

    <EXPORT NAME="netai_socket">
      <PREF NAME="FlowID">yes</PREF>
      <PREF NAME="ExportStatus">yes</PREF>
      <PREF NAME="ExportHost">localhost</PREF>
      <PREF NAME="ExportPort">4837</PREF>
      <PREF NAME="ExportProtocol">tcp</PREF>
    </EXPORT>

    <EXPORT NAME="ac_file">
      <PREF NAME="Filename">/home/nwilliams/example_output.txt</PREF>
      <PREF NAME="FlowID">no</PREF>
      <PREF NAME="ExportStatus">no</PREF>
    </EXPORT>

    <!-- export interval in seconds -->
    <PREF NAME="Interval">10</PREF>

  </GLOBAL>

  <RULE ID="1">
    <!-- match all udp/tcp packets -->
    <FILTER NAME="SrcIP">*</FILTER>
    <FILTER NAME="SrcPort">*</FILTER>
    <FILTER NAME="DstIP">*</FILTER>
    <FILTER NAME="DstPort">22,53,8000</FILTER>
    <FILTER NAME="Proto">tcp,udp</FILTER>

    <PREF NAME="auto">yes</PREF>
    <PREF NAME="bidir">yes</PREF>
    <PREF NAME="FlowTimeout">60</PREF>
  </RULE>
</RULESET>
```

There are several lines which are of interest.

```
<EXPORT NAME="netai_socket">
```

This configures the socket export features of NetMate.

```
<PREF NAME="ExportHost">localhost</PREF>
```

The host to which statistics are exported.

```
<PREF NAME="ExportPort">4837</PREF>
```

The netAI default port on which the `ExportHost` will be listening, which can be changed if desired.

```
<EXPORT NAME="ac_file">
```

Configure NetMate to write statistics to file.

```
<PREF NAME="Interval">10</PREF>
```

Will provide updates to the classifier at 10 second intervals.

```
<RULE ID="1">
```

Instructs NetMate about which flows should be exported (filters) and several other preferences. The various filters can be used to provide a more selective set of statistics, such as for particular hosts or ports. See below for an example of filter use.

```
<FILTER NAME="DstPort">22,53,8000</FILTER>
```

```
<FILTER NAME="Proto">tcp,udp</FILTER>
```

These filters were used to isolate ssh, dns and http flows passing through the capture interface. The IP protocols are restricted to TCP and UDP, as these are the protocols used by the applications.

Running the software

Now to see how to get everything up and running. First try running the netAI_CL.jar file independently of NetMate with the example files to get a feel for using the software.

Simple build and classify

This example will build a classifier from a training file and then classify some flows from a statistics file previously generated by NetMate. The results are written to file.

```
java -jar netAI_CL.jar weka.classifiers.trees.J48 -t example.arff -c 1 -A  
5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3  
6,37,38,39,40,41,42,43 -s example.stats
```

The command can be broken down as follows:

```
java -jar netAI_CL.jar
```

Just pointing the JVM to the netAI jar file.

```
weka.classifiers.trees.J48
```

This tells netAI_CL that we will be using the J48 algorithm for classification

```
-t example.arff
```

This is our training dataset.

```
-c 1 -A 5,6,7...
```

-c 1 indicates that the first value after -A is the class index (in this case 5). -A lists the attributes that you want to use from the data arriving from NetMate. These values are indexed from zero (0).

Generally you will want to load the classifier from a model file rather than training it from scratch. Depending on the selected machine learning algorithm and the size of the training file, training a new model can take a very long time!

Build and then classify from a statistics file

This example builds a classifier from a model file and then reads in a statistics file generated by NetMate. We want to see the destination port of the flow printed with the prediction, so the -Y 5 argument is used. If you look at the example.stats file, you will notice that the destination port of the flow is the 5th column (starting from 0). You can include other attributes by entering their column in comma separated form. The file is output using -o foo

```
java -jar netAI_CL.jar weka.classifiers.trees.J48 -m example.model -t example.arff -s  
example.stats -o foo -c 1 -A  
5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3  
6,37,38,39,40,41,42,43 -Y 5
```

You should see something like the following output:

```
netAI CL Version 0.1

Outputting features: 5
Output File: foo
Classification Algorithm: weka.classifiers.trees.J48

Model created in 0.1 seconds
Ready to receive from: example.stats
File Completed
```

Let's let at some of the new command issued here:

```
-m example.model -t example.arff
```

We use a model file to build the classifier this time. The ARFF training file is still included as it contains some information needed by netAI (in fact the `-t` command should be used in all cases).

```
-o foo
```

This time the output is written to a file called 'foo', which is more useful than dumping everything to the console. Its just a regular text file that you can view using your favorite text editor.

```
-Y 5
```

The Y switch allows us to specify what flow information is printed next to the flow prediction. This information is basically the same information specified with `-A`. Here we simply print out the destination port of the flow. Generally you might want to also see the source/destination IP addresses and ports.

Build your own model file

For convenience you can build a model from a training file using netAI. You can then use this model to perform testing in the future – without having to wait for training. We'll make a J48 model, and for kicks we'll include an algorithm specific parameter (unpruned tree). Remember, however, that whatever options you use when you build the model are permanent, that is you cannot change them at run time.

```
java -jar netAI_CL.jar weka.classifiers.trees.J48 -c 1 -t data/example.arff -B new-model.model
```

The main point of interest here is the `-B` option, which tells netAI that you want to build a model for the given filename. Don't forget to include the class index in the training file. The output will look something like this:

```
netAI CL Version 0.1

Training File: data/example.arff
Classification Algorithm: weka.classifiers.trees.J48

Building model from training file, ignoring other parameters
Model created in 0.91 seconds
Saving model to file: new-model.model
Completed
```

The `-B` option overrides all other options (aside from `-t`), so remember not to leave it in place when performing tests.

Build and then classify from tcp

This example builds a classifier from a model file and then waits for a tcp connection from NetMate. The predictions are printed to screen (you can always use the `-o` option if you want to classify to file). Use the `-a` option so that active flows are printed to the screen as well.

```
java -jar netAI_CL.jar weka.classifiers.trees.J48 -m example.model -t example.arff -c 1 -A
5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
6,37,38,39,40,41,42,43 -Y 5 -a
```

The output will look something like this:

```
netAI CL Version 0.1

Filename not specified, printing results to console
Outputting features: 5
Training File: example.model
Classification Algorithm: weka.classifiers.trees.J48

Model created in 0.89 seconds
Ready to receive from: TCP export
```

At this stage you will need to start NetMate. Assuming that the configuration file has been set correctly to export statistics via TCP, you might run NetMate using the following command (reads in from tcpdump file and exports via tcp):

```
netmate -f ~/dumpfiles/2005-12.06-vlan102.dump -r ../etc/netAI/netAI-rules.xml
```

You should then see some output similar to that shown below.

```
dns 1 53
dns 1 53
ssh 1 22
```

Here we have two DNS flows (dst port 53) and one ssh flow (22) correctly classified.

That's basically it for running the jar file directly. An easier (although with fewer options) way to run netAI is using the netAI script, and there are some examples below.

SCRIPT RELATED RUNNING INFO

Build and then classify from tcpdump file

This example builds a classifier from a model file, then uses NetMate to read in data from a tcpdump file for classification. The netAI rulset used when reading from file is: etc/netAI/netAI-rules-stats.xml. The results are saved into results.txt

```
./netAI start -f 2005-12.06-vlan102.dump -m example.model -t example.arff -A
5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
6,37,38,39,40,41,42,43 -c 1 -o results.txt
```

The output will look something like this before you are returned to the prompt:

```
Algorithm: weka.classifiers.trees.J48
Attributes:
5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
6,37,38,39,40,41,42,43
Class index: 1
Input: 2005-12.06-vlan102.dump
Output: prediction,probability,2,3,4,5
Starting netAI
Extracting flows and features...
Performing machine learning...
```

Note that there are several default values when using the netAI script. For example, by default the J48 classifier is used (as we have found this classifier to work well with our data).

Build and then classify from tcpdump file

This example builds a classifier from a model file, then uses NetMate to read in data from a network interface. The results are saved into results.txt. The netAI rulset used when reading from file is: etc/netAI/netAI-rules.xml. You will probably need to be superuser to perform this operation.

```
./netAI start -i eth1 -m example.model -t example.arff -A
5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
6,37,38,39,40,41,42,43 -c 1 -o results.txt
```

The output will look something like this before you are returned to the prompt:

```
Algorithm: weka.classifiers.trees.J48
Attributes:
5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
6,37,38,39,40,41,42,43
Class index: 1
Input: eth1
Output: prediction,probability,2,3,4,5
Starting netAI...
NetMate running with PID 3162
Classifier running with PID 3143
```

After returning the prompt, netAI will continue to operate in the background. The `netAI status` command can be used to check the PIDs while the software is running. You can view the classifications as they occur by using `tail -f results.txt`.

Stop the classifier using the command `netAI stop`.

That's all there is to running the script, as the syntax is very similar to using `netAI_CL`. Don't forget that the usage can be printed out using `netAI -h`, while the manpage also lists the options you can use.

NOTES

The example data provided with netAI is for introductory purposes and should not be considered suitable for use in real classification scenarios. In fact it is quite possible that when the supplied model is used with new data many of the classifications will be incorrect. For this tool to be useful you should use NetMate and Weka to produce datasets and models more specific to your needs. Using NetMate and Weka are outside the scope of this document, but learning to use these tools is strongly recommended.

Don't forget to check out the manpages for a description of each of the switches and options available.

ACKNOWLEDGEMENTS

This software has been developed with the support from Cisco Systems, Inc. under a University Research Program (URP) grant.

NetAI uses the WEKA (<http://www.cs.waikato.ac.nz/ml/weka/>) and NetMate (<http://www.ip-measurement.org/tools/netmate/>) software packages.

AUTHORS

Sebastian Zander (szander@swin.edu.au)
Nigel Williams (niwilliams@swin.edu.au)

COPYRIGHTS & LICENSE

netAI is released under the GNU public license (GPL) version 2. Please see the COPYING file included in the netAI package for details of this license.

Copyright 2005-2006 Swinburne University of Technology, Melbourne, Australia

netAI is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

netAI is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this software; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA