Data Model

The data sets used for training, testing have been created from Trace Files got from the University of Calgary project. As a part of that, 168 trace files were collected from 168 different users of Unix csh. There were 4 groups of people, namely *novice-programmers, experienced-programmers, computer-scientists, non-programmers*.

The model that we have tried to model is, that of a *novice-programmer* who has been gaining experience over the weeks and is gradually using more and more command sequences similar to that of an experienced user. This is a slow process that takes place over several weeks.

The data set we have created is as follows. There are 8 files, lets call them T0, T1, ... T7.

- T0 contains 574 commands from a *novice-programmer*
- T1 contains 567 commands from the *novice-programmer*, 30 from an *experienced-programmer*
- T2 contains 530 commands from the *novice-programmer*, 60 from the *experienced-programmer*
- T3 contains 532 commands from the *novice-programmer*, 90 from the *experienced-programmer*
- T4 contains 487 commands from the *novice-programmer*, 120 from the *experienced-programmer*
- T5 contains 448 commands from the *novice-programmer*, 150 from the *experienced-programmer*
- T6 contains 416 commands from the *novice-programmer*, 180 from the *experienced-programmer*
- T7 contains 396 commands from the *novice-programmer*, 210 from the *experienced-programmer*

While creating these files, care has been taken to ensure that if a sequence of commands were added into the file, then they are from complete sessions. Each session contains anywhere between say 2 and 200 command sequences. A sample command from the trace file looks as follows-

```
C pix assign3.p
D /user/cpsc211/l03b33/xxxxxxxe
A NIL
H NIL
X NIL
```

Here, the text following ‘C’ in the first line is the command executed. For the purposes of training and testing, the command executed has been used without the parameters.

For the training/testing phases, anomalous data has been injected. In our case, the anomalous data would exhibit behavior of an attacker/intruder. This has been simulated though use of command sequences got from *computer-scientists.*

There are two types of data sets that have been modeled, both contain the same structure as above, differing only in the way anomalous data has been injected. In the first case, 25 anomalies have been injected in every file from T1, through T7. In the second case, 25 anomalies have been injected only in the files T4 and T7. The motivation behind this is the belief that anomalous data is sparse and in real life, would be few and far between.

A sample sequence of commands from a file looks as follows -
[rwho , emacs, rm, rwho, f, jobs, f, f, jobs, cs, pascal, cat, emacs, pix, emacs, pix, ...]

A sample sequence of commands with anomalies in them (shown in bold) is as follows -
[mail, cd, ls, emacs, logout, cd, **echo, who, rwho, ls, emacs, lpr, lpq, size, ls, rm]**

Results

There are four types of training/testing that are analyzed -

- Static : Here, the training (creating the quantized dictionary) is done on file T0 and, the quantized dictionary created at this point is used in finding anomalous patterns in files T1..T7
- Most Recent : Here, the algorithm is implemented such that , at the 'i'th iteration, training is done in file Ti and testing on file Ti+1
- Dynamic : This is similar to the previous case except that, when we train on a new file, the previous quantized dictionary is not discarded, the new quantized dictionary is built upon the existing one.
- Dynamic (With Enhancements) : The idea behind this model was that, when anomalies are detected during testing phase, it would be preferable that they are not used as training data in the next phase. Hence, the detected anomalies are shown to the user and, information about the compromised sessions are fed

In the graphs shown below,

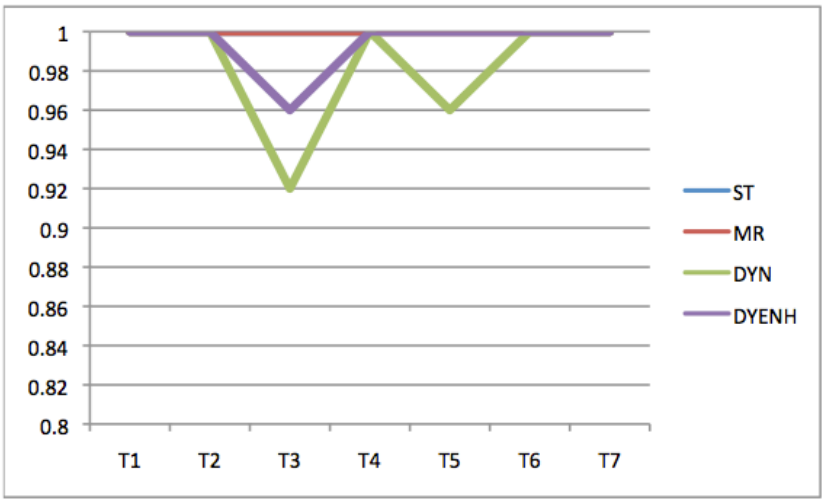
ST stands for Static training
MR stands for most recent training
DYN stands for dynamic training
DYENH stands for dynamic training with enhancements mentioned above.

x-axis corresponds to the file being tested on, T1 through T7 in each case

y-axis in each case, corresponds to either the true positive rate (TPR) or false positive rate (FPR) , depending on the graph.

Scenario where there are anomalies in files T1 through T7

TPR

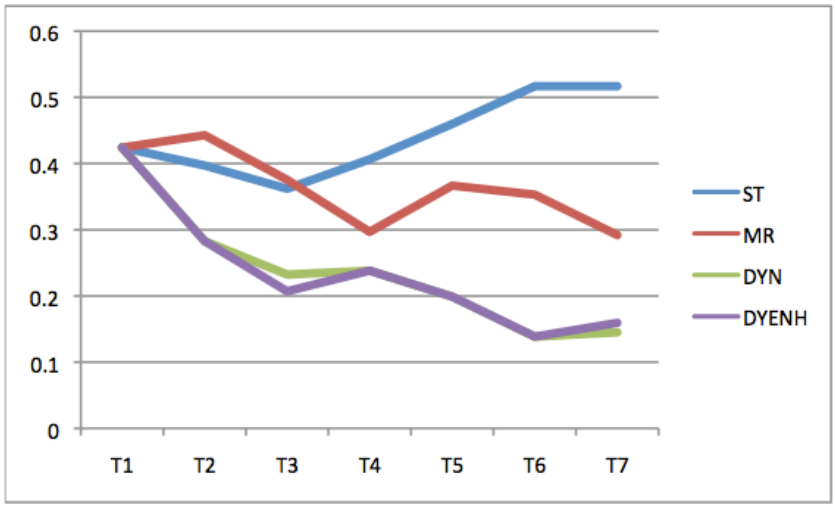


The MR, ST training give a TPR value of 1 every time.

Dynamic training goes down to 0.92 TPR at the most but compensates with a significantly lower FPR as we shall see.

For this statistic (TPR), we see that the anomaly detection rate is high in all cases.

FPR



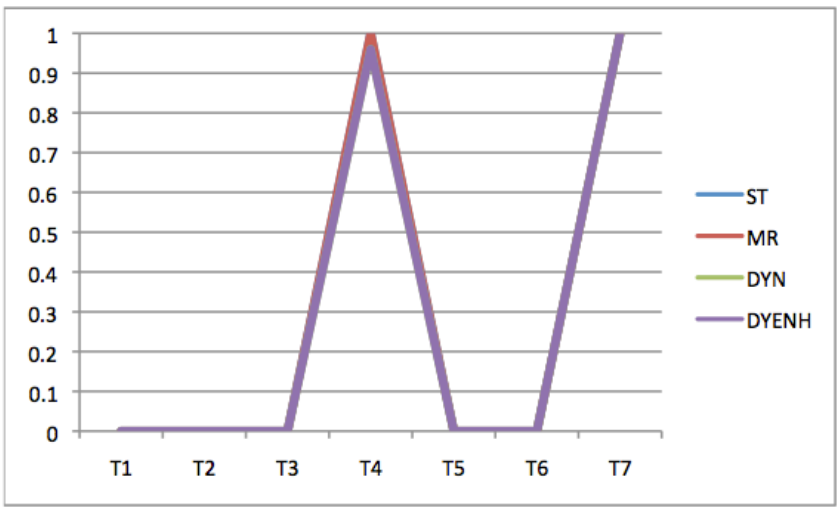
The FPR value is higher for ST, MR as can be seen.

The DYN & DYENH values provide the best performance.
At T6, it reaches the best FPR value:-
ST 0.516
MR 0.353
DYN 0.1384
DYENH 0.1384

For this statistic (FPR), we see that the number of false positives decreases with more training (in the case of dynamic). This is majorly due to the fact that the dynamic training accounts for the concept drift in a much better manner than the other techniques.

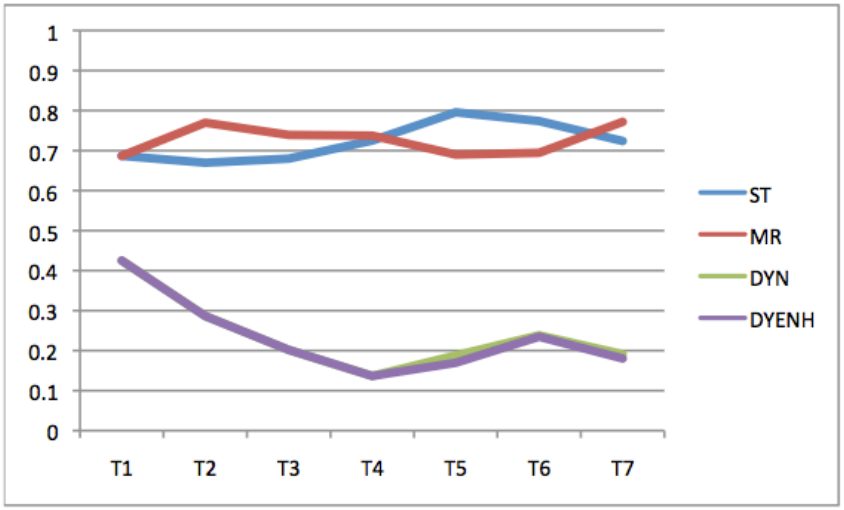
Scenario where there are anomalies in files T4 and T7

TPR



In this case, the TPR values are highest when tested on T4, T7 and give a value of 1 in both cases.

FPR



The FPR values decrease gradually for DYN, DYENH, best FPR value is seen at T4 :
ST 0.7248
MR 0.7374
DYN 0.1367
DYENH 0.1367

In this case, the distinction between dynamic and static training is more profound.