# Programming HW 2

Fall 2013
Shlomo Hershkop
3137 Honors Data Structure

Out: Oct 30
Due: Nov 26th (roughly)

## *Programming Section (50 points)*

This project involves the use of a hash table to identify a unknown program whether it is malicious or not. **You don't need to fully understand the Security background in order to do the assignment, but it won't kill you to learn about new things** ☺

**Security background**

Most security systems (network protection programs aka IDS, and binary protection programs aka virus checkers) in use today are based on digital signatures. This means that some expert has sat down (for lots of money) and used some tools to encoded a (hopefully unique) signature of bad programs into a database. The signature is a given sequence of one and zeros to represent a unique malicious program or malicious toolkit. Using these signatures, programs such as Norton Anti-virus, Mcaffee, etc are able to identify programs which are viruses and then pop up a little window saying your computer is infected. (which most people just click ok and don't bother reading)

In short: given a program on your machine, your anti-vurs will read through it and compare its binary data to the virus binary data it has stored (hence the need for updating the virus databases).

The advantage of this approach is the low false positive rates, that is if the program thinks something is bad, it is probably true. This is because the signatures contain enough information, that the computer has effectively "memorized" what the virus looks like. The main drawback of the signature-based approach is the inability of identifying new viruses which don't reuse portions of old viruses. In addition, this approach also has to deal with the vulnerability window. This is the time it takes for a new signature to be created, tested, and updated to the main computers that use them. In the real world this window can be quite large in comparison with the spread of viruses. Especially since many viruses spread via emails. Those viruses are known as 'zero day' virii since they take advantage of an unknown problem until someone (say a company rhyming with nikrosoft or snapple, as a chance to fix it.)

Using Data Mining and Machine Learning algorithms it is possible to create a model of what a virus looks like without necessarily memorizing them, and use this learned model to identify programs which might be viruses.

One of these ways is to use n-byte sequences of a virus program to create a dictionary of what a size n byte sequence of a virus would look like. Then, calculating the probability of specific byte sequences of being malicious. For example, if you look at a million virus programs and a million good programs, and notice that the sequence 1111-0001 only occurs in the virus programs, you can say with high probability that if you see it in an unknown program it has a high chance of being a virus (remember we don't base our prediction on one byte sequence….its a simplified example).

A byte sequence does not have to be a single byte. A 'window' is defined as a sequence of n bytes. So a n=1 window, would examine each byte, while n=3 window would look at every 3 bytes together. The tradeoff is that as we increase the window, we are in affect 'memorizing' since our data points will shrink. That is, smaller window sizes with the same number of examples, tend to give you better accuracy since you see more example of each type.

You will be given a few known virus programs (v) and a few normal programs (denoted b= benign). You will build a general dictionary data structure (aka hashtable) which can take both virus and benign byte sequences and create a dictionary to be able to score an unknown programs.

You will write a java program which will allow a user to add good and bad programs to the malicious/benign database and to save/load such a database in order to check a specific program to see the probability of it being good or bad.

When the program starts it should display a guy to ask the user what they want to do, the actions available:

1) Designate the dictionary directory
    a. This is the specific directory where you store your DS. This allows you to have multiple dictionaries if you want to play around with your program
    b. This will also in effect 'load' a previous session if it exists.
2) Clear all probability information
    a. Empty the hash tables to start from scratch
3) add information to benign file
4) add information to virus file
5) calculate probability of unknown file
    a. should ask use for the file and output the probability of being malicious and probability of being benign…and should predict what it is.
6) information
    a. this should provide some info about yourself, and how many files have been added to the system
7) quit
    a. ask use to save the dictionary file if necessary
    b. stop running your program

We can use a naïve bayes approach to calculate the probability of any byte sequence of being malicious or benign. The probability of a byte being malicious can be computed as #times seen in malicious set divided by the total number of times seen in any set. The probability of being benign is the #times in benign set over all times seen. The method assumes conditional independence between byte sequences, which allows you to simply multiply the probabilities to get total probabilities (or simply sum the logs).

You will need to package your assignment into 2 sets: the src java files in a tar.gz file and a binary .jar file which the TAs will be able to click on and execute.

Please start early and ask for help. We will cover specific parts of the assignment in class, but it will still take you some time to put the pieces together and debug the assignment.

please sketch out the pieces on a paper and come see us if you do not understand specific pieces or don't have an idea on where to start.

Last: Please have fun and get creative….if you would like pointers on where to find live virus programs stop by office hours. please budget extra time to cleanup your machine if it gets infected :)