REPORT:

The first thing I did was to download the given files and extract the relevant information. This was done in Wireshark by filtering with “frame.len==102&&ip.src==192.168.56.102”. This was done on each file, and the results were exported as csv.

I then made the function parseCSV.m that formats the csv-files in a more sensible manner. This includes removing all irrelevant fields and placing the relevant ones in a matrix.

When estimating the unknown typings, I took a “the more the merrier”-approach. I created several solutions using different techniques before including them all in a “super solution”.

The first solution is the function bestMatchEuclid.m. This takes the timestamps of an unknown command as input and calculates the Euclidean distance to each of the 20 known command-typings we were provided. Euclidean distance is a common way to calculate the differences between different vectors and is often used due to its simplicity (Wang, et al., 2005). If the lowest distance is found in the grep-set, it assumes the unknown command to be grep, and sudo if the lowest distance is found in that set. This method was able to correctly identify all the known command-typings. This is due to that there was always one of the matches where the Euclidean distance was 0 because the input was one of the known command-typings that it was matched against. It is not certain that all would be estimated correctly otherwise.

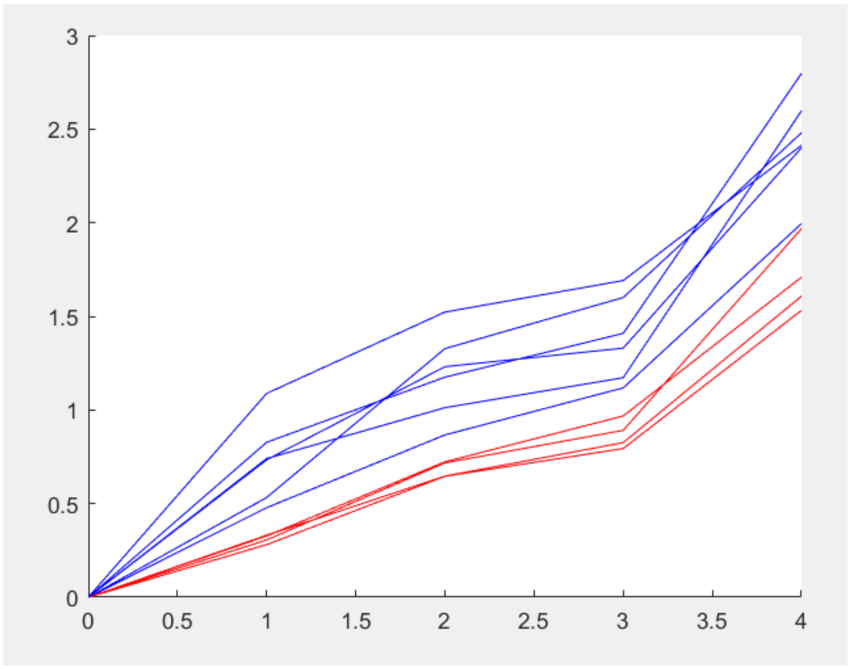
The second solution is the function averageEuclid.m. This also used the Euclidean way of measuring distance, but instead of matching the input with each of the 20 known command-typings, the input was matched with the average command-typing of the grep- and sudo command-typings. The average command-typing was calculated using mean() in MATLAB. This method was able to correctly guess 90% of the known grep-typings and 60% of the known sudo-typings. When estimating the sudo-typings almost half of the estimations were wrong, which is not much better than random guessing.

The third solution is the function bestMatchManhattan.m. This works exactly as bestMatchEuclid.m, but instead of comparing the Euclidean distance, the comparison is made using Manhattan distance, which is another common way of measuring distance between different vectors (Chang, et al., 2009). This method was able to correctly estimate all the known command-typings. Although, due to the same reasons as in bestMatchEuclid.m, there was always a match with Manhattan distance equal 0.

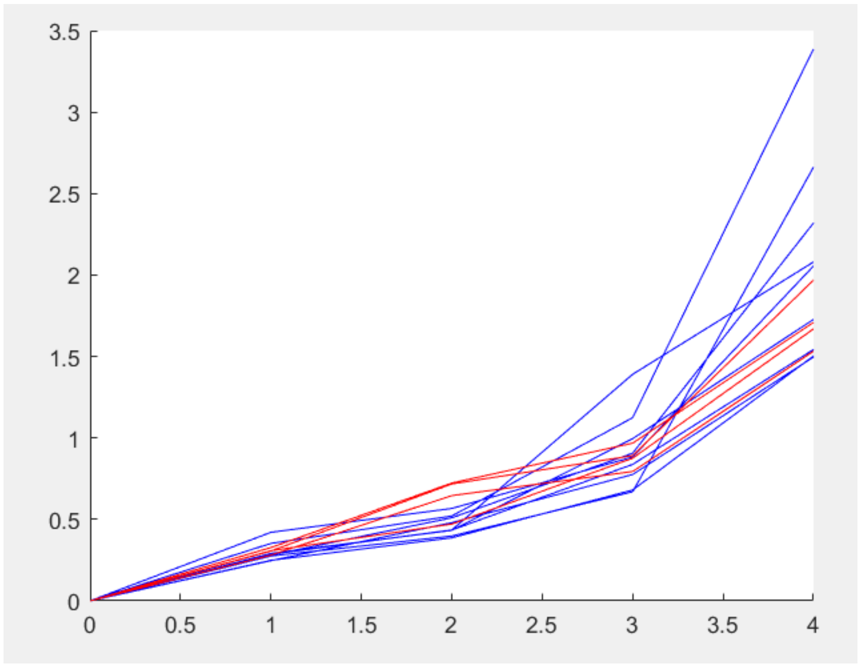
The fourth solution is the function averageManhattan.m, which is the same as averageEuclid.m except for using Manhattan distance instead of Euclidean distance. This method was able to correctly guess all grep-typings, but only 60% of the sudo-typings.

The fifth and sixth solution are bestMatchChebyshev.m and averageChebyshev.m. These follows the same patterns as the previous ones, only measuring distance using Chebyshev’s way (Klove, et al., 2010). The former correctly estimated all known typings, but the latter correctly estimated 80% of the grep-typings and 60% of the sudo-typings.

An interesting observation is that averageEuclid.m, averageManhattan.m and averageChebyshev.m, which all only guessed 60% of the known sudo-typings correctly, failed on the exact same sudo-typings (Number 1, 4, 6 and 9). I therefore plotted the sudo-typings as seen in the graph below:



The blue lines represent the sudo-typings averageEuclid.m, averageManhattan.m and averageChebyshev.m guessed correctly, while the red lines are the sudo-typings they guessed wrong. They were likely missed because they are too much below the average. I plotted these red lines together with the grep-typings as well, which can be seen below:



It seems that these particular sudo-typings fits better with the grep-typings. More data is needed to be certain, but I think that if almost half of the sudo-typings are closer in distance to the average grep-typing than the average sudo-typing, makes averageEuclid.m, averageManhattan.m and averageChebyshev.m too unreliable for estimating which command is typed.

The seventh and final solution is timingBetween234.m. When manually looking at the known typings I noticed that for sudo-typings, the space between the 2nd and 3rd timestamp was bigger than the space between the 3rd and the 4th timestamp. And for grep-typings, the space between the 3rd and 4th timestamp was bigger than the 2nd and 3rd timestamp. I think the reason for this is that the physical location of the keys on the keyboard. 'E' is much closer to 'r' than 'p' and 'd' is closer to 'u' than 'o'. As this was true for all the known typings we received, I feel that it can be said with some certainty that if the space between the 2nd and 3rd timestamp is bigger than the space between the 3rd and 4th timestamp, it is more probable that command was sudo rather than grep. This solution correctly estimated all the known typings.

I then combined all these into the super solution main.m. All unknown typings are ran through all the solutions. Each solution gives a point to either grep or sudo. After all solutions have given their opinion, the final estimation is the one with the most points. Due to the somewhat poor results of averageEuclid.m, averageManhattan.m and averageChebyshev.m, it could be wise to leave them out of the super solution, but due to the number of other solutions, the final results did not change based on they were included or not. I therefore decided to keep them in.

The following table displays how each of the different solutions and the super solution estimated the unknown typings:

bestMatchEuclid.m	averageEuclid.m	bestMatchManhattan.m	averageManhattan.m	bestMatchChebyshev.m	averageChebyshev.m	timingBetween234.m	Super solution (main.m)
Sudo	Grep	Sudo	Grep	Sudo	Sudo	Sudo	Sudo
Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo
Grep	Grep	Grep	Grep	Grep	Grep	Grep	Grep
Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo
Grep	Grep	Grep	Grep	Grep	Sudo	Grep	Grep
Grep	Grep	Grep	Grep	Grep	Grep	Grep	Grep
Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo
Grep	Grep	Grep	Grep	Grep	Grep	Grep	Grep
Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo
Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo	Sudo

There is little deviance between the different methods, so I think I can be somewhat confident in the result of the super solution.

Instructions:

To run the software, one only needs to run main.m, no parameters needed. To experiment with different combinations of methods, one can change the "methods"-vector which contains function handles to all the created solutions.

References

Chang, D.-J., Desoky, A. H., Ouyang, M. & Rouchka, E. C., 2009. Compute Pairwise Manhattan Distance and Pearson Correlation Coefficient of Data Points with GPU. I: *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing.* s.l.:IEEE, pp. 501-506.

Klove, T., Lin, T.-T., Tsai, S.-C. & Tzeng, W.-G., 2010. Permutation Arrays Under the Chebyshev Distance. *IEEE Transactions on Information Theory,* 56(6), pp. 2611-2617.

Wang, L., Zhang, Y. & Feng, J., 2005. On the Euclidean distance of images. *IEEE transactions on pattern analysis and machine intelligence,* 27(8), pp. 1334-1339.