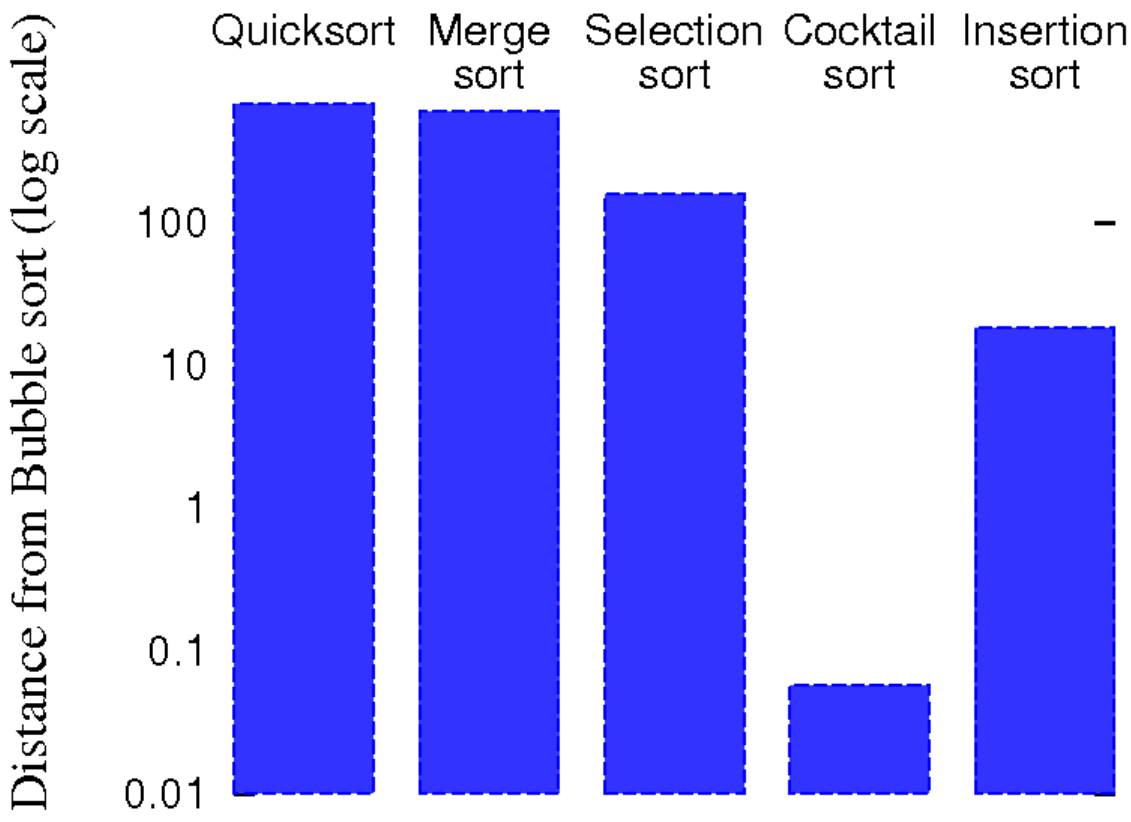
Dynamic Information Flow Tracking (DIFT) software release - Java version

This is a GPL release of software developed as a prototype for performing Dynamic Information Flow Tracking (DIFT) with vectors as the taint marks. The software was developed by Antonio Espinoza.

The taint marks being vectors has two benefits:

1. It is possible to quantitatively track information flow for challenging dependencies, including address and control dependencies.
2. It is possible to compare the provenance of any two pieces of information simply by comparing the two vectors associated with them (*e.g.*, using a dot product).

For instructions on using the software see the README file. If you have any questions about the software, please email crandall@cs.unm.edu.



Right now the software can trace any statically-compiled Linux binary to produce an annotated assembly trace, and then use that annotated assembly trace to build a branch correlation matrix. In this matrix, the rows are branches in the program and the columns are inputs to the program. The elements of the matrix are a sum over time of the correlations between the data that influenced the branch decision and all inputs, i.e., every time a branch decision is made the taint of the control flag the branch is predicated on is compared to every column in the matrix with a cosine similarity and this value is added to the corresponding element. So the elements of the matrix can be interpreted to mean: for the execution trace of the program, how much influence did each input have on each branch decision in aggregate.

As an example of an application of this form of dynamic information flow tracking, refer to the figure above. Here we compare the matrices for sort programs by taking their outer product scaled to a number between -1.0 and 1.0 by their 2-norm. This is then converted to a distance. It can be seen in the figure that Cocktail sort (a variant of Bubble sort) is very close to Bubble sort, while Quicksort and Merge sort are far away.

To generate the trace of a static Linux binary, the ptrace interface is used to single-step the program and the assembly for each program counter location is extracted using binutils. In future releases the Linux binary can have dynamic linking, but for this release the Linux binary must be statically linked.

Whenever input is read in on specific file descriptors (the first file opened by default), all words of input are tainted with random vectors. The length of each vector represents the amount of taint, and the direction has no meaning except with respect to other vectors. The vectors are given many dimensions so that any two input vectors are orthogonal to each other with high likelihood. As tainted information propagates through the program, the taint marks are also propagated. Taint vectors are combined by adding the vectors and scaling back the vector's length, so that the new taint mark has correlation with both sources of the combined information. The branch correlation matrix is built as the program runs by updating the matrix every time a conditional branch instruction occurs.

The new version of this software is written in C and based on a standard binary disassembly framework so that the sources of x86 traces are more flexible. It will also model the desctruction of information over time, something not modeled in this version.

This Java version is no longer maintained, contact us at the email address above for information about the release of the C version.

This software was developed with support from the Defense Advanced Research Projects Agency CRASH program under grant #P-1070-113237.

This work is also supported by the National Science Foundation under grant #1017602.