Design Brief: Computer Security Visualisation

Group 4 - 21 February 2013

- **What we have been up to.** In several meetings, we discussed the assignment and possible approaches to implementation. In particular, we considered which kinds of data input we want to use, how to analyse it, and different ways to visualise the data. For the latter, our aim is to give an observer an intuitive overview of patterns in the data with the possibility of adding on filters to see more detail. Further we talked about ways to modularise our programming approach to facilitate cooperation.
- **Where we are now.** We want to develop a Java-based application which makes use of a third-party graphics library to display visualisations of a user-specified dataset, in such a way that it makes it easy for the end-user to notice ongoing or previous network attacks. We want our visualisations to show the current state of the network, that is updated in real time. For implementation purposes, we plan to simulate time passing on pre-collected network dumps. This will also make it possible to analyse past network activity, and allow the user to speed up or slow down time.
- **Initial ideas for visualisations.** Our main ideas are displaying a map of nodes in the network, and providing charts of the data going to each IP. The map should show which clients are most active, and what kinds of traffic are used, and use statistical information calculated from past states of the network to visualise variance. In particular we want it to be possible to filter by type of packet, and by host. We could use colour coding, tag clouds, and different sizes to mark data. Further information can be displayed using plain text in Swing components.
- **Our plans going forward.** Develop common interfaces for visualisation, filters, data handlers, and data input. Familiarise ourselves with the JOGL system, and how to combine this with Swing interfaces. Familiarise ourselves with network packets, network dumps, and how to interpret input data, using Wireshark or Snort. Find efficient ways of dividing tasks among group members according to individual strengths, and setting up ways of collaboratively developing code using `git`.