Group Report: Computer Security Visualisation

Clockwork Dragon Group - Group 4 - 30 April 2013

- Our application consists of a series of five visualisations of network traffic, each of which offers a different representation of the data. These are designed to enable a user to detect a variety of potential cyber-attacks. We have also constructed 'filters' which allow the user to limit the visualisations to show only a selection of the data according to their individual needs.

  All of our visualisations are drawn in real-time, i.e., they are updated as new data gets available. While the application now focuses on analysing pre-collected data, it is easily extendable to support visualising network traffic as it happens.

  Here are brief descriptions of our visualisations:

  HeatMap, TrafficVolume, Ports,

  DataFlow: Saturation indicates time (the gray lines are past lines) the coloured lines are more recent. All the lines are transparent therefore if a line stands out it means lots of packets go through that line. The red triangles show traffic. They shrink with each iteration by some percentage and grow linearly with each packet on that interval. They also have an upper limit.

  Cube

  If the user wants to analyse only a selection of the data, it is possible to filter the input data by over 400 protocols (such as TCP, HTTP, etc), by port, and by IP address using whitelists and blacklists. The main advantage of the filter system is that it works application-wide and applies to all visualisations.

  The application in its current state uses packet captures in CSV format[1]. The data includes IP address, MAC address, and X for both sender and destination, and also information on port and protocol used, and the size of the packet data. The packets listed in the CSV file are then 'simulated' to arrive in real-time. The user can choose to slow down or speed up the simulation, to pause it, or to skip to the end of the data capture.

  The application also includes an Analysis Panel to provide real-time statistics of the processed data packets.

- During Hilary term, we met several times as a group to discuss the assignment, and possible approaches to implementation. We produced an initial specification outlining what the final product was supposed to achieve. We then converted this design brief into a list of common interfaces we needed to design, and of required features to implement. We then each selected tasks to work on, proceeding to split them into smaller tasks as it became necessary. Later, we migrated to GitHub, and used the service's issue management.

[1] see appendix for a specification of the file input format

During the vacation, we kept in touch and discussed the progress of the project using a combination of GitHub (to track bugs, comment on each other's code, and as version control) and Google Hangout (to plan further development). It was during the vacation that most of our work was done. Particularly helpful was the prompt development of the back-end of the application which is responsible for processing the input data and making it available to the visualisations, but also to display background data to the user.

After this, we focused on producing a number of visualisations to illustrate the data. During the entire development phase, the use of GitHub facilitated collaborative and concurrent development to a great extent.

Before the development phase, we were unsure about how to allocate work to people in the group. We didn't see a very natural splitting. While we did assign responsibilities for data collection, data processing, and visualisations to different people, we kept the working area of each person flexible so as to allow their input on all parts of the application. This proved to be a valuable decision. In particular, since there was a kind of linear structure to the tasks (data has to be collected before it can be processed, and it has to be processed before it can be visuablised), this informal approach to splitting tasks meant that development proceeded faster than it might have done otherwise.

- Reflecting back on the things we learned from this project, there are on the one hand a list of technical tools each of us needed to become familar to: git and GitHub, JOGL, and an understanding of the workings of computer networks. More importantly, we faced the troubles but also opportunities that collaborative software development brings with it, and understood the importance of a clear modularisation of code, provision of easy-to-understand interfaces, and good documentation, so that other people can quickly familiarise themselves with and work on other's code.

  In retrospect, we had a somewhat slow start to our project. This was because many of us were unfamiliar with JOGL and networks in general which meant that it took a while of familiarisation with the basic concepts to be able to think up ideas for the program and be creative in our visualisations. During this time, we were helped by the variety of resources that our sponsor made available to us.

- Looking at the application in the state it is now in, we can see a number of ways it could be extended. Live processing of data would obviously increase real-world usability. There are also ways in which the filter system could be made more powerful. Finally, by using more statistical (pre-)processing of the input data it could be possible to render more insightful visualisations.