Birdseye

Executive Summary

Purpose of Project

The purpose of our project is to give users a network visualization of the traffic that enters/leaves their home computer. By accomplishing this, our objectives are two-fold: to educate users, and to allow users to see where suspicious packets, if any, are going on a live map.

Firstly, Birdseye is intended to be educational. The application will give users a better understanding of how data is routed within a network. Through this, the hope is to encourage users to learn more about computer networking in general, and bring up a new generation of network engineers.

The second intention of Birdseye is to give users an in-depth look at where their packets are going. This is useful for users who believe their packets are being intercepted or going to suspicious locations. Using Birdseye, users can get an indepth look at exactly where their packets are traveling, and the hops the packets are making along the network.

Design

Birdseye infrastructure follows the Model-View-Controller (MVC) architecture. Both the front and the back end use this architecture. There are three major components to Birdseye: a packet capture utility (done in C), a traceroute utility and web-server (done in Python), and a graphical frontend (done using web technologies).

- Web based project
- MVC on frontend (with backbone), backend (with flask)
- restful apis
- multiple languages (C, python, javascript
- Multi-process & multithreaded
- 100% client side code.
- Front end code uses the observer pattern to full effect, where the views “listen” to changes on the models, and re-render themselves to constantly reflect the current state of the application.

Verification

Birdseye utilizes a comprehensive test suite. It includes unit tests that cover both back and front-end functionality, style tests and integration tests. The majority of these tests are set to run automatically on a pull request to the git repository, ensuring that they are run on every iteration of the build.

- Front/back end unit tests include
    - Python unittest, and Javascript QunitJS tests.
- Back end style tests
    - According to Python PEP8 standards.
- Automatic builds on pull request/Integration tests
    - Implemented using TravisCI.

In told, the back-end packet capture utilizes 12 tests with a 100% pass rate.

GeoIP utilizes 5 tests, covering all reasonable situations with a 100% pass rate.

All in all, our tests cover all reasonable situations the user will encounter, and all tests pass with a 100% success rate.

Conclusion

Birdseye is a great tool that not only educates users but also gives them an additional to to further analyze their network traffic. Students, youngins, and network admins would benefit from using Birdseye, even if only as a novel application to play with. Birdseye is written in widely known languages using design patterns that makes it easily reusable and extensible. Thus, it would be easy for future users/fans to come along and add new features to Birdseye. Currently, Birdseye is a well tested application with an easy to use UI that works great right out of the box.