Building a Web Based Search Application for SiLK Data

Author: Cayler Miley, Undergraduate; Researchers: Cayler Miley, Daniel Lopez; Mentors: Jeff Springer, Nancy LaTourette
Computer Science and Engineering Department; University of Nevada, Reno

ABSTRACT

Monitoring a large-scale network requires a robust application to query enormous amounts of data and present search results with readability for security analysis. Building an application to find and report on security events from many network gates presents problems involving efficiently querying massive databases and presenting lists of IP addresses, ports, and other network data in a way that allows the application user to effectively process security events with a high threat level in real time.

INTRODUCTION

- The project arose to monitor IP conversations on a complex network.
- We set up a web service based on “a collection of traffic analysis tools developed by the CERT Network Situational Awareness Team (CERT NetSA) to facilitate security analysis of large networks” referred to as a SiLK database for tracking netflow information.
- Python scripting and a microframework called Flask helped complete a server side setup where a URL could be called to query the SiLK database.
- A GUI was needed to present the data from the SiLK database for readability and ease of use for end users. To achieve this we used a single page web application that accepted JavaScript Object Notation (JSON) for the results from the web server.
- These components form a web application that can aptly determine the network location and time that security events occurred and further research trends in cyber attacks on the network.
- This application inspired us to create a video presentation to interest local students in cyber security through explanations of phishing attacks.

PROCESS

- In this presentation, the process of creating an application that can handle large scale searches for unwanted communication on a network and present the data it returns is completed through the use of a web service and a browser interface.
- Our developmental approach was to use a python framework called Flask to interface a query from a Graphical User Interface (GUI) to the System for Internet Level Knowledge (SiLK).
- The GUI offers an easy way for users to search on all of the conversations a particular set of IP addresses has had during a set time interval and sort through large sets of data to determine if a security event needs to be created based on the information provided by the web service.
- The process and logical use for an application of this kind offers an excellent way to interest local students in cyber security, which has resulted in two videos on the cyber security topics for presentation at local high schools.
- The development of the application is still continuing and is scheduled for production testing in May of 2016.

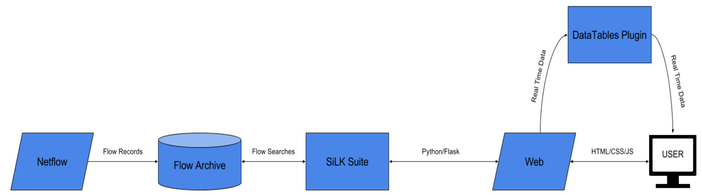
METHODS

- The server side web service (referred to as the “back end”) uses a custom Python module to operate on the SiLK database and manipulate the data returned from a query. The python framework Flask is used for URL management.
- Flask routes URL strings to python functions allowing a user to activate python scripts from a web browser with access to the domain name of the server containing the back end. We used this to easily pass parameters in a URL to the python functions and allow all of the computation required by SiLK to complete on the server side.
- During implementation, threading and forking solutions were used and tested for efficiency and speed of computation. The time taken for searching the SiLK database was large enough that a status function was required so that the user could check if their query was still working on the database. Once finished, the data is returned.
- The user interface accesses Flask through a URL in a web browser (Google Chrome, Mozilla Firefox, etc.).
- The GUI (referred to as the “front end”) was implemented using HTML, CSS, and Javascript. To better present the data, the DataTables plugin was used for graphical representation of the data.
- The Ajax framework allows the user to dynamically recompose the web page based on their new searches to make the application single page. We used this to limit the need for refreshing the page.
- The page is built on Google Material Design Lite (MDL) to make an appealing and easy to understand interface.
- As development continues, we have encountered problems with a way to inform the user of the status of a large search and to optimize the search process for the web service.

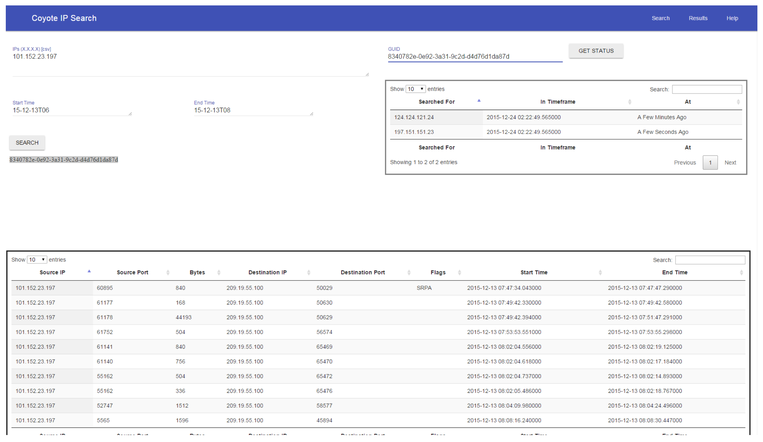
RESULTS

- Development thus far has resulted in a single page web application that displays tabular data relevant to network security events, called Coyote.
- The single page GUI uses a web form to route data to Flask which can then run a python script to query the SiLK database. The data response on the query then populates a table using the DataTables plugin.
- The front end is still under development to include features such as a loading progress bar or icon while the data table loads, accepting universal time intervals to be re-formatted for a Flask route on the client side, and a table of recent searches to allow a user to effectively receive previous data without re-running a search.
- Both forking and threading solutions were tested but threading produced significantly faster results on searches of more than 3 IP addresses at a time. Even so, a status function for the is required as some searches can take minutes.
- Each search is uniquely identified by a Globally Unique Identifier to allow users to not repeat identical searches. This serves as the component for querying the status of the SiLK database.
- The back end of the application is relatively robust and free of major bugs, thus is likely ready for production. The front end is still in development with production testing for the entire application scheduled for May 2016.

RESULTS



Current Implementation of the Coyote Architecture.



User interface for the Coyote application.

CONCLUSIONS

- The Coyote application was developed as a way to monitor network security events on large-scale networks to prevent and/or respond to attacks on the network.
- The Coyote application allows for efficient and timely response to events with its easy to use GUI, fast database searching, and graphical presentation of data.
- The single components of the Coyote application offer monitoring a network in close to real time when events occur.
- The project has not encountered any major overloads in data or crashed due to user input and data overload, therefore it looks to function as a production level monitoring application with horizontal scalability to meet the needs of its user base.

This material is based upon work supported in part by the National Science Foundation under Grant No. IIA-1301726.

Note: All data presented on this poster was anonymized to limit possible security breaches while the application is under development.