**Overview**

LCARS is a web-based security application designed to identify, analyze, respond to, and help prevent attacks and threats targeting network infrastructure. Using this diagram as a starting point, we divided LCARS into three categories: Analysis, Threat Intelligence, and Threat Response, which we call the Reconfigurator.

**Data Collection**

We collect live, raw, attack data from LongTail SSH honeypots and BlackRidge gateways. When data comes in we parse it into JavaScript Object Notation (JSON). This allows for straightforward integration with our relational and graph-based analytics tools. It also gives us the ability to handle data coming from multiple sources. The screen shots below show raw and parsed data from one of our honeypots. Here, parsing gives us the source IP address and the attempted username and password for each attack.

**Threat Intelligence**

We built a Threat Intelligence database of attack profiles, response recipes, and orchestrated responses. A response recipe is a collection of firewall rules. An orchestrated response maps an attack profile to a group of response recipes. To interact with this database we built our own REST API. Our API enables us to easily create, update, and delete database items directly from our GUI.

**Analysis**

We currently take our parsed attack data and generate hive plot visualizations, G* graph commands, and SQL commands. The graph and SQL commands are sent to BiG* Data Studio, a front end to the G* graph database and PostgreSQL. This allows us to execute our graph and SQL commands and run queries against them. The hive plot pictured on the left represents the same attack data as the graph and SQL on the right.

We plan to automate this whole process using logs from live routers and SDN controllers so LCARS will dynamically reconfigure the network if it detects an attack.

**Reconfigurator**

The Reconfigurator enables deployment of response recipes and orchestrated responses to our firewall. We utilize RFW (Remote Firewall), an open-source REST API for iptables, in order to seamlessly interact with the firewall service. Firewall rules can be deployed both in batch or on an individual basis.