Using the Siren Platform for Superior Business Intelligence

by Bill Wicker | Mar 20, 2019

Can we use a single platform to uncover and visualize interconnections within and across structured and unstructured data sets at scale?

Objective

At Risk Focus we are often faced with new problems or requirements that cause us to look at technology and tools outside of what we are familiar with. We frequently engage in short Proof-Of-Concept projects based on a simplified, but relevant, use case in order to familiarize ourselves with the technology and determine its applicability to larger problems.

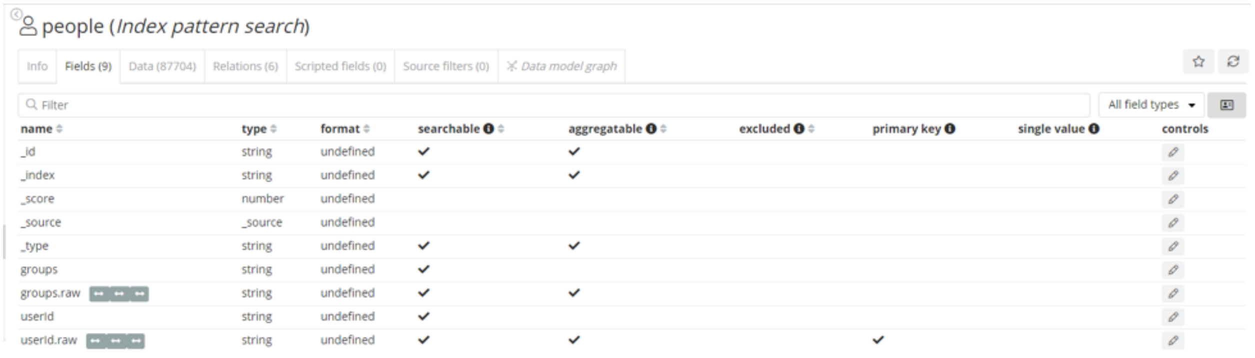
For this POC, we wanted to analyze email interactions between users to identify nefarious activity such as insider trading, fraud, or corruption. We identified the Siren platform as a tool that could potentially aid in this endeavor by providing visualizations on top of Elasticsearch indexes to allow drilling down into the data based on relationships. We also wanted to explore Siren's ability to define relationships between ES and existing relational databases.

The Setup

Getting started with the Siren platform is easy given the dockerized instances provided by Siren and the Getting Started guide. Using the guide, I was able to get an instance of the platform running with pre-populated data quickly. I then followed their demo to interact with Siren and get acquainted with its different features.

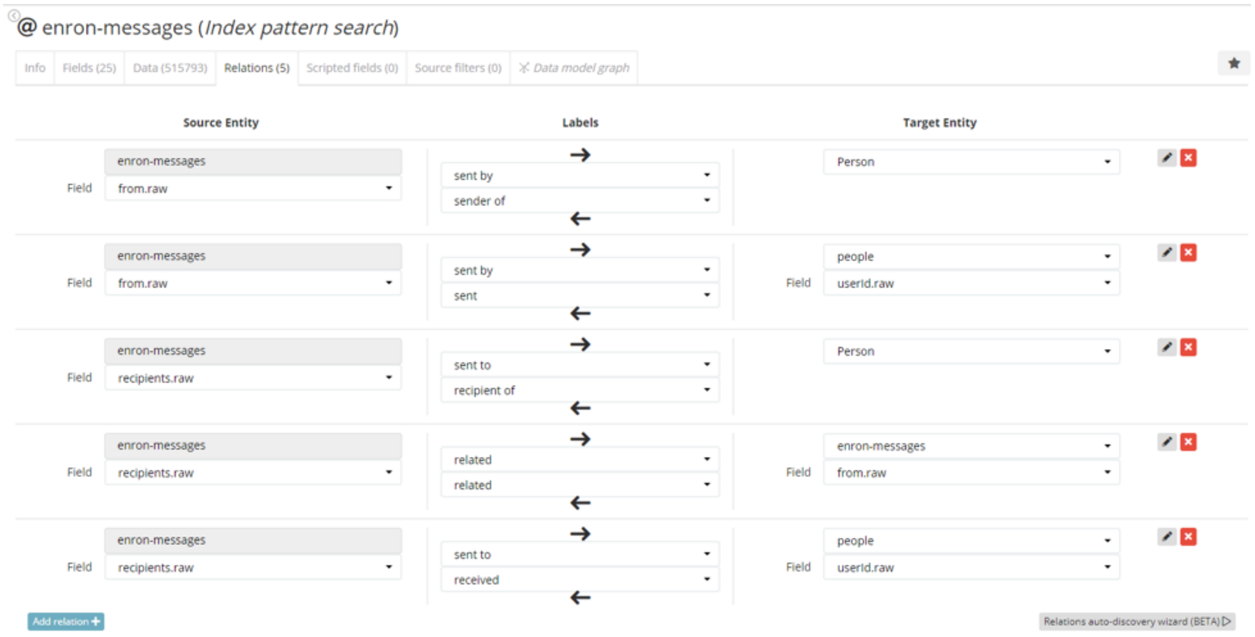
Once I had a basic level of comfort with Siren, I wanted to see how it could be used to identify relationships in emails, such as who communicates with each other and if anyone circumvents Chinese firewall restrictions by communicating through an intermediary. I chose the Enron email corpus that is publicly available as my test data and indexed it in ES using a simple Java program that I adapted from code I found here. I created one index containing the emails and another index of all the people, identified by email address, who were either senders or recipients of the emails. The resulting data contained half a million emails and more than 80,000 people.

With the data in place, I next set up the indices in Siren and defined the relationships between them. The straightforward UI makes this process very simple. The indices are discoverable by name, and all of the fields are made available. After selecting the fields that should be exposed in Siren, and potentially filtering the data, a saved search is created.



name	type	format	searchable	aggregatable	excluded	primary key	single value	controls
_id	string	undefined	✓	✓				
_index	string	undefined	✓	✓				
_score	number	undefined						
_source	_source	undefined						
_type	string	undefined	✓	✓				
groups	string	undefined	✓					
groups.raw	string	undefined	✓	✓				
userId	string	undefined	✓					
userId.raw	string	undefined	✓	✓		✓		

Once all of the indices are loaded and defined, the next step is to define the relationships. There is a beta feature to automatically do this, but it is not difficult to manually setup. Starting with one of the index pattern searches, the Relations tab is used to define the relationships the selected index has to any others that were defined. The fields that are used in the relationships must be keyword types in ES, or primary keys for other data sources.

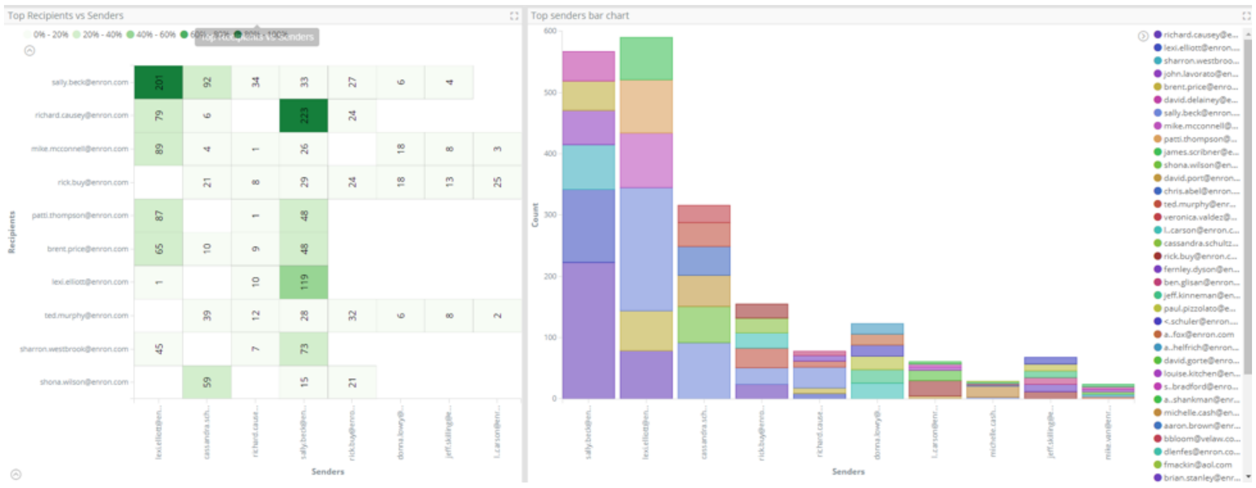


Source Entity		Labels		Target Entity	
	enron-messages	→	sent by		Person
Field	from.raw	←	sender of		
	enron-messages	→	sent by		people
Field	from.raw	←	sent	Field	userId.raw
	enron-messages	→	sent to		Person
Field	recipients.raw	←	recipient of		
	enron-messages	→	related		enron-messages
Field	recipients.raw	←	related	Field	from.raw
	enron-messages	→	sent to		people
Field	recipients.raw	←	received	Field	userId.raw

Now that the indices are loaded and connected by relationships, the next step is to create some dashboards. From the Discovery section, the initial dashboards can be automatically created with a few common visualizations that are configured based on the data in the index. Each dashboard is linked to an underlying saved search which can then be filtered. There is also a visualization component that allows for filtering one dashboard based on the selection in a related dashboard.

Dashboard

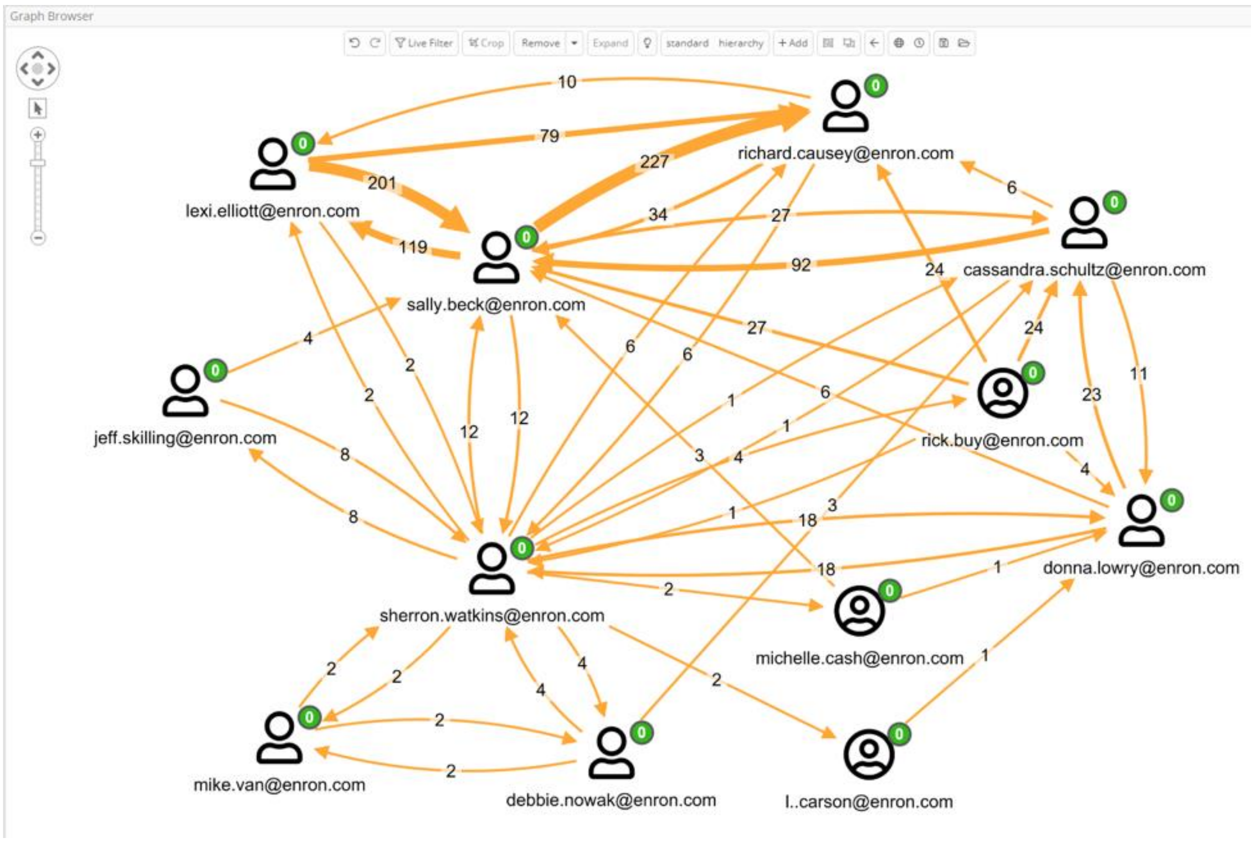
Each dashboard is typically associated with a saved search and contains different visualizations based on the search results. Some of the visualizations show an aggregated view of the results, while others provide an alternative way to filter the data further and to view the results. Once the user has identified a subset of data of interest in a particular dashboard, s/he can quickly apply that filter to another related dashboard using the relational navigator widget. For example, one can identify a person of interest on the people dashboard and then click a link on the relational navigator to be redirected to the email’s dashboard, which will be filtered to show just the emails that person sent/received.



The above screenshot shows two versions of the same information. The people who sent the most emails are on the x-axis, and the people they emailed are on the y-axis with the number of emails sent in the graph. By clicking on the graphs, the data can be filtered to drill down to emails sent just from one person to another, for example.

Graph Browser

One of the most interesting features of Siren is the graph browser, which allows one to view search results as a graph with the various relationships shown. It is then possible to add/remove specific nodes, expand a node to its neighboring relationships and apply lenses to alter the appearance of the graph. The lenses that come with Siren allow for visualizations such as changing the size or color of the nodes based the value of one of its fields, adding a glyph icon to the node, or changing the labels. It also supports custom lenses to be developed via scripts.



In the screenshot above, I started with a single person and then used the aggregated expansion feature to show all the people that person had emailed. The edges represent the number of emails sent. I then took it one step further by expanding each of those new nodes in the same way. The result is a graph showing a subset of people and the communication between them.

Obstacles

As this was my first foray into both Elasticsearch and Siren, I faced some difficulty in loading the data into ES in such a way that it would be useful in Siren. In addition, the data was not entirely clean given that some of the email addresses were represented differently between emails even though they were for the same person. There were also many duplicate emails since they were stored in multiple inboxes, but there was no clear ID to link them and thus filter them out.

Apart from the data issues, I also had some difficulty using Siren. While the initial setup is not too difficult, there are some details that can be easily missed resulting in unexpected results. For example, when I loaded my ES indices into Siren, I did not specify a primary key field. This is required to leverage the aggregated expansion functionality in the graph browser, but I didn't find anything in the documentation about this. I also experienced some odd behavior when using the graph browser. Eventually, I contacted someone at Siren for assistance. I had a productive call in which I learned that there was a newer patch release that fixed several bugs, especially in the graph browser. They also explained to me how the primary key of the index drove the aggregated expansion list. Finally, I asked how to write custom scripts to create more advanced lenses or expansion logic. Unfortunately, this is not documented yet, and is not widely used. Most people writing scripts just make modifications to the packaged ones currently.

Final Thoughts

In the short amount of time that I spent with Siren, I could see how it can be quite useful for linking different data sets to find patterns and relationships. With the provided Siren platform docker image, it was easy to get up and running. Like with any new technology, there is a learning curve to fully utilize the platform, but for Kibana users this will be minimal. The documentation is still lacking in some areas, but the team is continuously updating it and is readily available to assist new users with any questions they may have.

For my test use case, I feel that my data was not optimal for maximizing the benefits of the relational dependencies and navigation, but for another use case or a more robust data set, it could beneficial. I also did not delve into the monitoring/alerting functionality to see how that could be used with streaming data to detect anomalies in real-time, so that could be another interesting use case to investigate in the future.