Project Final Report

GIS Data Visualization of Firewall Traffic

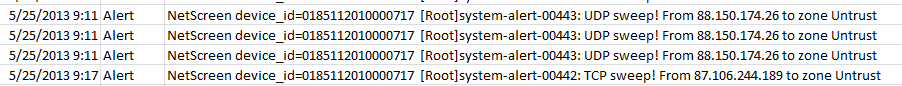
|  |
| --- |
| Juliana Lo |
| Advisor : Dr. Michael Thomas |
| Presented at the ESRI User Conference on July 22, 2015  Penn State University |
|  |

# **Introduction**

Firewall is a security device that sits on organization’s perimeter network stopping unauthorized access to inside network. Data traversed through the firewall are tracked in log files, information captured including originating IP addresses, port number, time stamp, type of traffic, and severity level. Firewall captures two types of traffic: normal traffic such as web surfing and emails, and malicious attack traffic, traffic from hackers looking for vulnerabilities to break into the network. The goal of this project is to develop an interactive web application to visualize the bad traffic requests. The events captured have no coordinate system, only the IP addresses, but the data can be mapped into latitudes and longitudes using a geolocation database, saved to a SQL database to be used for display and analysis. The project has several objectives we want to accomplish:

* Create geolocation records from IP addresses. Visualize data base on severity and frequency.
* Display the information close to near real-time.
* Make use of Software as a Service (SaaS) cloud computing infrastructure for cost savings.
* Develop a feasible workflow for data capturing, geolocation, and data loading.

# **Data Set**

The firewall data comes from a Juniper Network’s NetScreen firewall. Below is a screen capture of firewall’s GUI console displaying event time, device id, alert level, protocol, and source IP address.  
  


# **System Design Methodology**

The project’s development process is described below, from requirement analysis, design, implementation, to testing.

**Requirement Analysis**  
Requirement analysis is the first stage in the development cycle. I gathered inputs from IT system staff about their needs for a mapping solution, discussed project goal and objectives, and identified high-level system capabilities. Once all the requirements were collected and analyzed, a preliminary conceptual design was created using website mockup and wireframe tools. We agreed on key components on what the application will look like, performance measures for real-time data, and a basic test plan validating system at end of the project.

**System Design**  
  
In this phase of development life cycle the entire system was defined in detail and each component described. The system was divided into modules to make it easier more manageable. Document hardware required, system configuration changes, software to be developed, and any other resource required to complete the system. Key activities were:

* Evaluated several GIS cloud vendors based on database technology, ease of use, application programming interface, and pricing. CartoDB was selected because the company offers a cost-effective cloud-based solution based on PostGIS database, and a robust programming interface. Paid for an inexpensive starter plan at $29 monthly with 100 MB space quota, unlimited private datasets, dashboard publishing capability, and programming API.
* Researched Internet literature on how to capture firewall traffic to external machine, decided on using syslog transports. Syslog is a standard for computer data logging, defined in RFC 3164/3195 and uses UDP/TCP transport on port 514. It’s a way for network devices to send event messages to a logging machine, usually known as a syslog server. Evaluated several syslog software packages, SnmpSoft Syslog Watcher was selected for the project mainly because there is a freeware version, free but still can collect syslog information for up to 5 firewalls.
* Researched geolocation methodology. The most widely used technique for geocoding consists of building a database to keep the mapping between IP blocks and a geographic location. A quick survey found several commercial databases (IP2Location, GeoIP, GeoLite, etc.) on the market. They all have a “lite” edition which is free, but with limited accuracy & number of records, and all have application interface in several programming languages. MaxMind’s GeoLite’s database was selected for this project. The database is free of charge, updated once per month, database format is in both binary and CSV formats.
* One of the key decisions for implementation stage was the choice of programming language for the web application. I evaluated several languages and decided on Python and Javascript. Python is a strong scripting language, easy to write and integrates well with vendor’s application programming interface. Javascript is heavily used today for web development. It is supported by all all browsers and works on most mobile platforms. I also researched Javascript libraries Google Chart and jQuery for use in dashboard components.

The application development framework is divided into five separate modules:

* Data Collection – capture traffic requests from the firewall.
* Parsing Engine – software module to extraction the information needed for geocoding (IP address, time stamp, protocol, alert levels).
* Geolocation Service – software module to translate IP addresses into locations (city and country).
* Database Service – database module to append new geospatial features.
* Visualization – program to create data visualization maps.

System hardware and software components list:

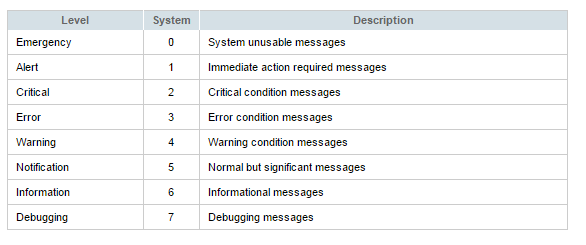
* Firewall - the first component is a hardware firewall already exists on the network. Its model and make is a Juniper Netscreen firewall.
* Database – cloud-based CartoDB database where the records will be stored.
* Integrated Development Environment (IDE) - a windows 7 development machine for syslog service, parsing engine, geolocation service, and database service.
* Map application – browser-based application written in Javascript, jQuery, CartoDB SQL, and Google Charts libraries.

**Coding, Testing, and Implementation**  
  
Coding is the phase where software development began to take shape, the development framework modules took approximately six weeks to complete. The parsing, geolocation, and database modules were written in a combination of Python script and APIs, visualization program written in Javascript language. Each module was tested individually, assembled together for integration and end-to-end testing, and tested for different errors by using various test cases and in different browsers (Chrome, Firefox, and Internet Explorer).

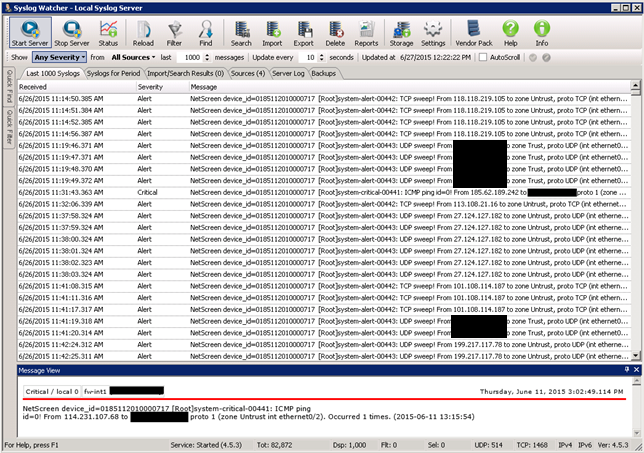
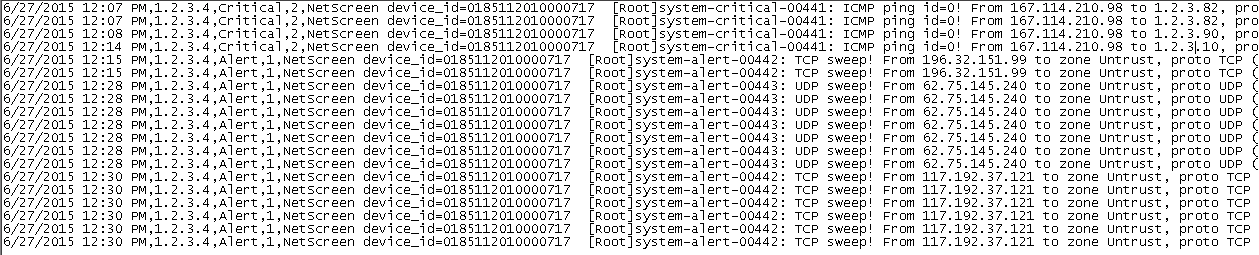
# **Workflow Process**

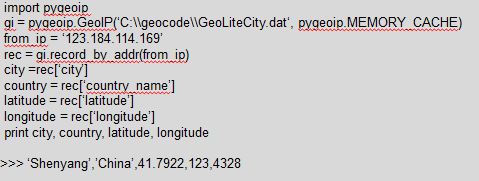
The workflow describes the overall data flow from beginning to end.

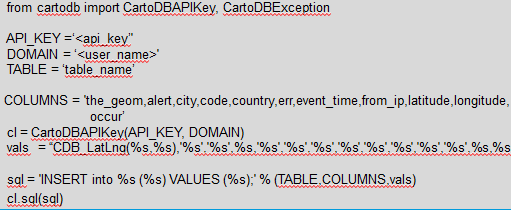
**Step 1: Configure firewall for data capture.**  
There are eight firewall log event levels in descending order of severity (figure 1). I configured the firewall to capture on the most severe events: critical, alert, and emergency. Its setting was also changed to send log events to an external syslog server.

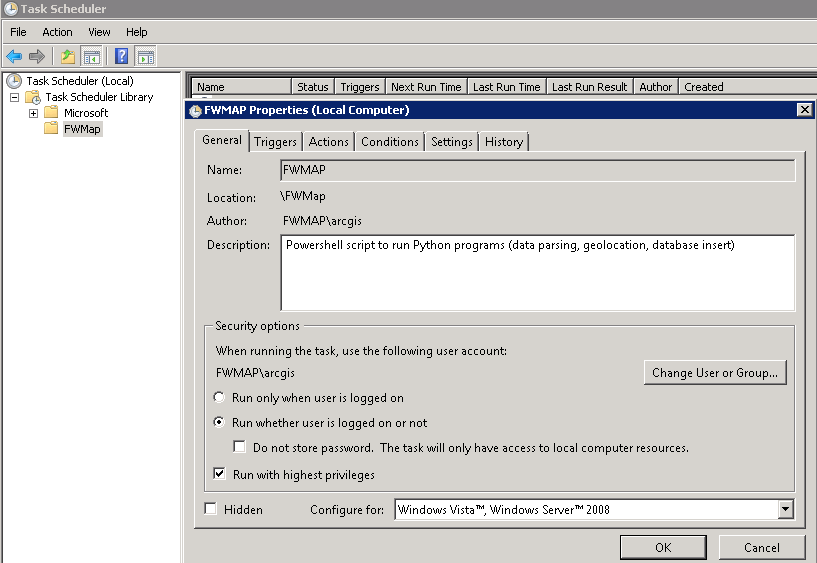
Figure 1 shows firewall event levels.

**Step 2: Syslog Server**  
SnmpSoft’s Syslog Watcher software (figure 2) was installed on the IDE machine. Software setting modified to export new records to a flat file in CSV format based on time intervals.

  
  
Figure 2 shows a screen capture of Syslog Watcher GUI console.  
 **Step 3: Parsing Engine Module**  
Parsing engine process data (figure 3) exported from syslog server in step 2. I wrote a simple data extraction script in Python language to extracts time stamp, firewall host, message level, type of error, the IP address where the traffic came from, port number, and number of hits.  
  
  
  
Figure 3 shows data exported from syslog software.

**Step 4: Geolocation Engine Module**  
In geolocation module the IP addresses are matched to geographic locations. I am using a database-based look-up method based on MaxMind’s GeoLite database; location accuracy is limited to city and country level. The database has Python API (figure 4) with just a few lines of codes, input parameter is an IP number, returns results are the name of city, country, latitude, and longitude.  
  
  
Figure 4 shows geocoding process, pygeoip is MaxMind’s python API.

**Step 5: Database Module**  
The data is appended to database. CartoDB also has a Python API which is pretty simple to implement, inserting features by constructing raw SQL queries.   
  
   
Figure 5 shows database insert statement

**Step 6: Automate the Process**  
Workflow is not complete without automating the steps, this is crucial because one of the project objectives is to support near real-time monitoring. To do this I use Windows Task Scheduler program to run a PowerShell program calling Python scripts (parsing engine, geolocation, and database) every 5 minutes.  
  
  
Figure 6 shows Window’s Task Scheduler

# **Results**

The resulting application is a dynamic viewer showing firewall traffic mapped to geographic locations, maps that change with underlying data.

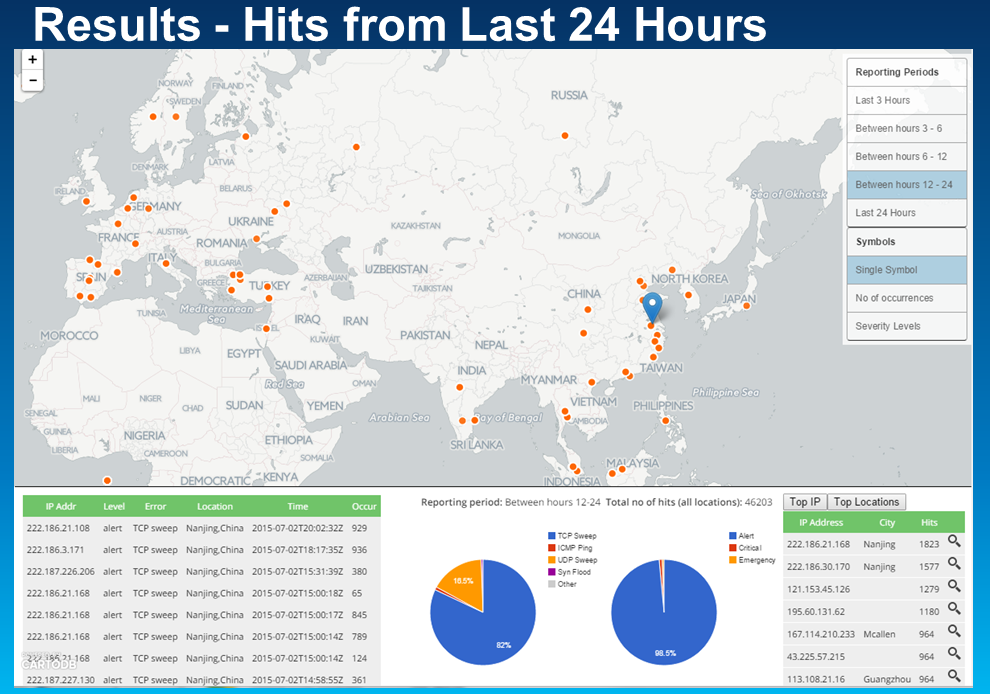
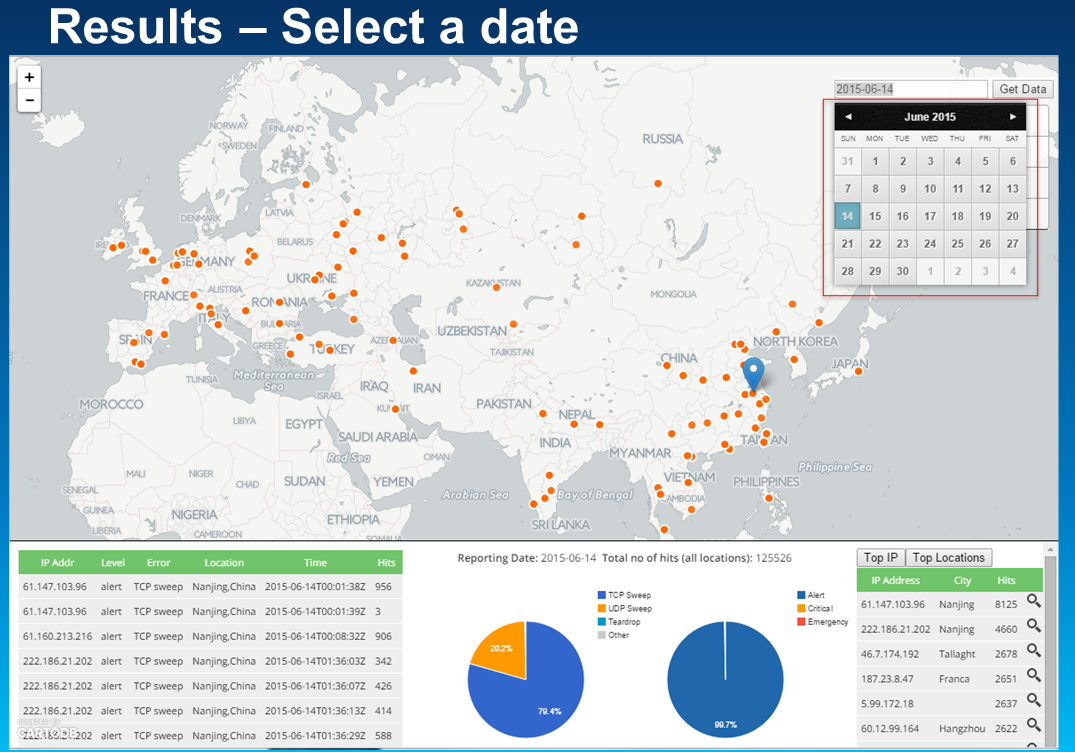
  
Figure 7 shows firewall traffic from last 24 hours.

Figure 7 shows one of the maps displaying traffic hits from the last 24 hours. The map window is divided into 3 areas: layer/symbol selector area where a user can select from different reporting periods and symbol styles; mapping area with a simple background base map and geolocated IP addresses; and a dashboard area showing tables and charts. The application implemented clickable features. When a user clicks on a feature in the map, the application will up the requests came from that location and display the results (IP address, error, time stamp, and number of hits) in dashboard area (table on the left). The application also tabulates the top hosts or cities having the most hits in a given period. The results are shown in the table on the right, click on the zoom-to button to navigate to the location. The Pie charts are constructed from Google Charts API. They show the distribution of hits by error types and by severity levels.

The reporting periods are not limited to last 24 hours. In a separate map (figure 8) the calendar function allows the user to select a date, the application searches the database for records for the date selected and returns the results on the map.

  
Figure 8 shows firewall traffic from a previous date  
  
For additional analysis I created a choropleth map (figure 9) showing the number of hits by countries, joining database records to country borders, counting the number of hits from each country.

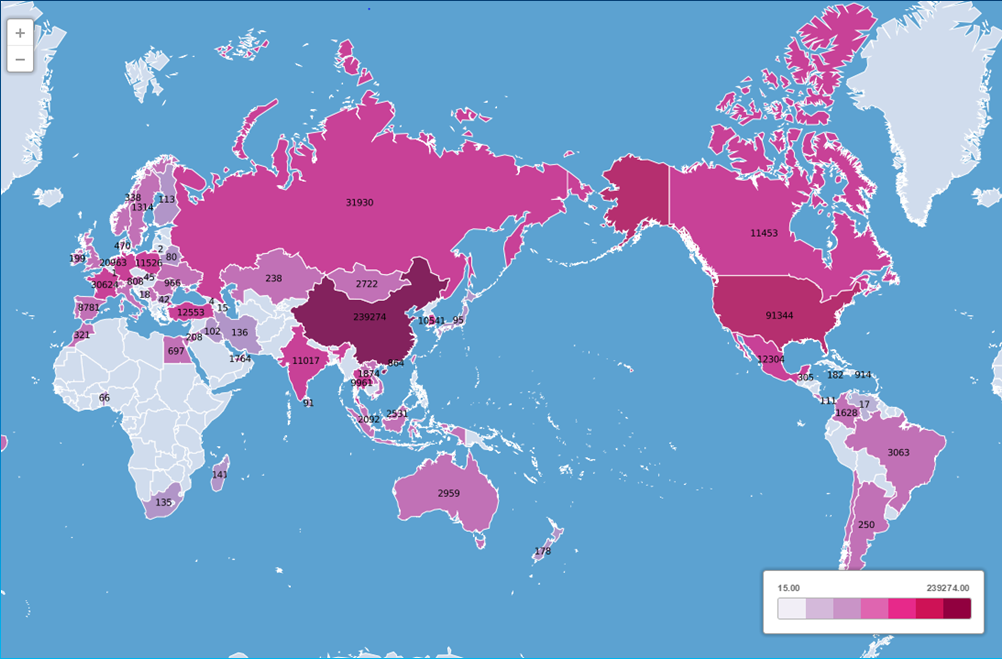
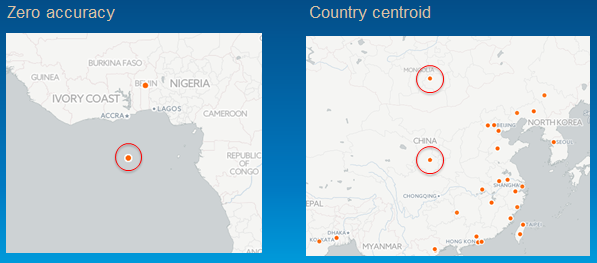


Figure 9 traffic hits from each country

There are additional screen captures in the Power Point presentation including temporal time animation and pie charts.  
  
**Geolocation Issues**MaxWind geolocation database accuracy varies from country to country. Accuracy on a city for the US within a 50 km radius is 67%, comparing with China which has a low accuracy of 38%. In some cases the locations either has zero accuracy (0.0 latitude, 0.0 longitude) or mapped to country centroid (figure 10).

  
Figure 10 geolocation accuracy

One of the limitations of IP based geolocation is that often times, services returns the geographic coordinate of the server on which that IP is connected to the Internet, not necessarily the location of user's machine. There are also other ways of hiding actual IP addresses behind proxy and VPN service.

# **Conclusion**

This project demonstrates that GIS can be used to visualize firewall events, use geolocation technology to map locations of IP addresses. The process of capturing firewall requests, parsing data, geolocation of IP addresses, database updates, and automation make up a real-time workflow to deliver timely information to IT department staff. The live, dynamic maps produce useful intelligence; identify patterns that could help improve monitoring activity and decision making. The project leverages GIS cloud platform for development, Software as a Service (SaaS) model is cost effective for a small to medium project without having to stand up my hardware and software infrastructure. CartoDB’ subscription model is easy to understand, service is quick to setup and produce visualizations, geospatial features stored in CartoDB’s PostGIS database, and API to help create web maps. The location accuracy from MaxMind’s database is a concern. It will be interesting to compare results between free and paid databases. More research is needed on improving accuracy and detecting proxies.