Summer Internship Report

Written by:	Ron Gorai
Company:	Kovid Inc.
Industry:	Cybersecurity/Analytics

Introduction

During the summer of 2017, I worked at Kovid Inc. as an intern on a new cyber-analytics project. The goal of this project was to detect suspicious activity in user data from the NEICE servers.

NEICE stands for the National Electronic Interstate Compact Enterprise. It is a cloud-based system that is used to place children across state borders, managed by the Interstate Compact on the Placement of Children (ICPC). This system substantially shortens the time it takes for this process and saves a hefty amount of money by cutting the need for printing, copying, and mailing documents. NEICE launched in November 2013 with six states signed on (DC, FL, IN, NV, SC, WI), with a goal of reaching all 52 US territories by May of 2018.

Client

Our client for this project, American Public Human Services Association (APHSA), is the primary sponsor for the NEICE system. In addition to working with clients in Health and Human Services, Kovid also works with clients in the Corrections and Public Safety.

Business Need

With any program like this, the agencies always have a concern for the safety of the children. Hence, the agency has a responsibility to detect illegal activities or malpractices such as data theft, fake data uploading, placing of children in random places, etc. Kovid was tasked by APHSA to create necessary tools to detect suspicious users, ensure the welfare of the children and possibly even save lives in the long run.

Project Overview

My project has two parts: automatic and manual detection.

For the automatic method, my colleague, Aneesh, and I took a set of data from the NEICE servers, parsed it into a format that was easy for us to use – deleting unnecessary data – and programmed a module that implements support vector machine kernels and various sub-processes. This part of the program uses the kernels to automatically recognize unusual activity based on usual activity and then creates reports for us to see.

For the manual method, we created visualizations from the parsed data. The manual detection is a secondary process, enabling a human to visually detect any outlying data that the machine may have missed.

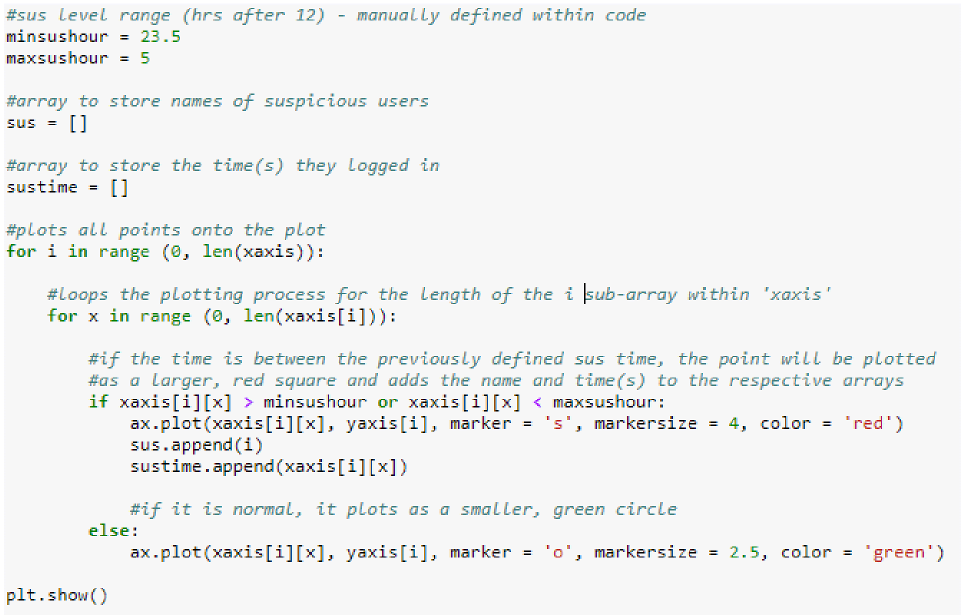
My Contributions

My role in this project was to develop the visualizations of the parsed data. I started the summer with a mini project in which I took a set of random IP addresses and traced their locations using an external API from *tools.keycdn.com*. I used this exercise to get myself familiar with Python and to get gist of what we were going to do moving forward. Afterwards, Aneesh and I made a program to generate test data. We then used this test data to make a build a parsing program since we didn't have access to the actual data at this point.

Once we received the actual data from the server logs, we split up the work load: Aneesh worked on the machine learning aspect of the program while I worked on the visualizations. The first step was perhaps the most difficult; I had to determine what parts of the data were necessary for the visualizations.

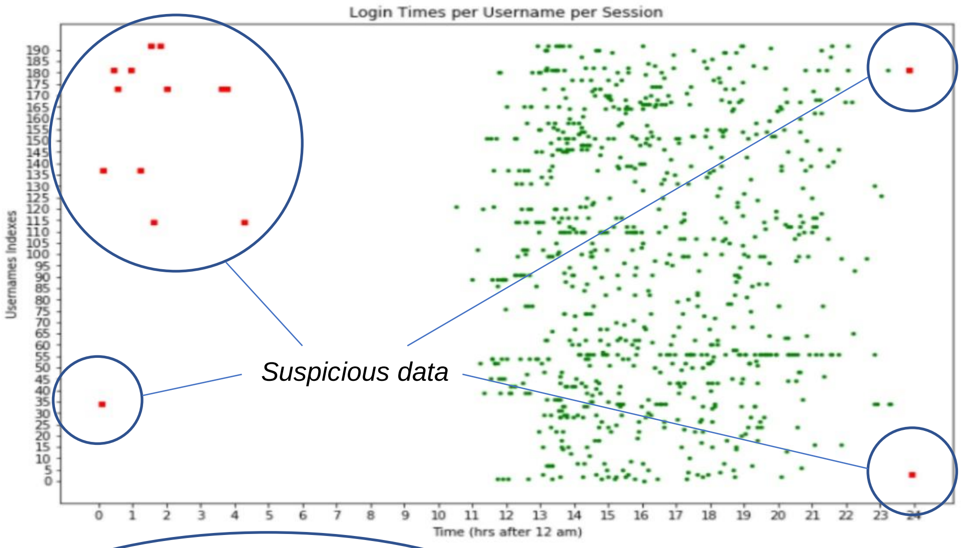
Visualizations

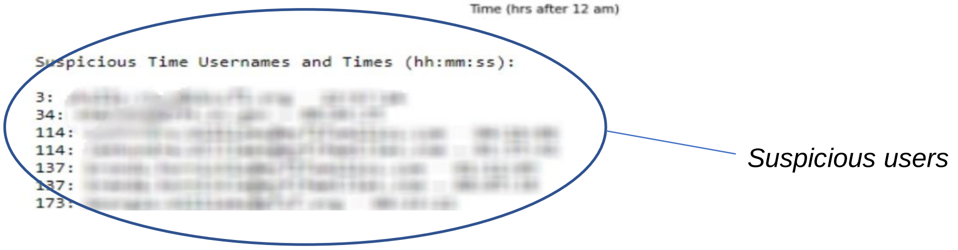
The first visualization I created was a chart of login times. I made this in the form of a scatterplot, so I could plot the login times of user for each session they are online in a given timeframe. I also implemented a feature where the chart plots login times deemed suspicious (these values are manually defined within the program) as larger, red squares rather than the smaller, green circles of the other data. The names of users who logged on during the suspicious time frame are then displayed along with the time they logged on. The objective of this visualization was to see if people were logging in at odd hours, like 3 AM for example.



```
#sus level range (hrs after 12) - manually defined within code
minsushour = 23.5
maxsushour = 5

#array to store names of suspicious users
sus = []

#array to store the time(s) they logged in
sustime = []

#plots all points onto the plot
for i in range (0, len(xaxis)):

    #loops the plotting process for the length of the i sub-array within 'xaxis'
    for x in range (0, len(xaxis[i])):

        #if the time is between the previously defined sus time, the point will be plotted
        #as a larger, red square and adds the name and time(s) to the respective arrays
        if xaxis[i][x] > minsushour or xaxis[i][x] < maxsushour:
            ax.plot(xaxis[i][x], yaxis[i], marker = 's', markersize = 4, color = 'red')
            sus.append(i)
            sustime.append(xaxis[i][x])

            #if it is normal, it plots as a smaller, green circle
        else:
            ax.plot(xaxis[i][x], yaxis[i], marker = 'o', markersize = 2.5, color = 'green')

plt.show()
```

This excerpt of code from the login times chart illustrates the process I used to plot each point onto the chart

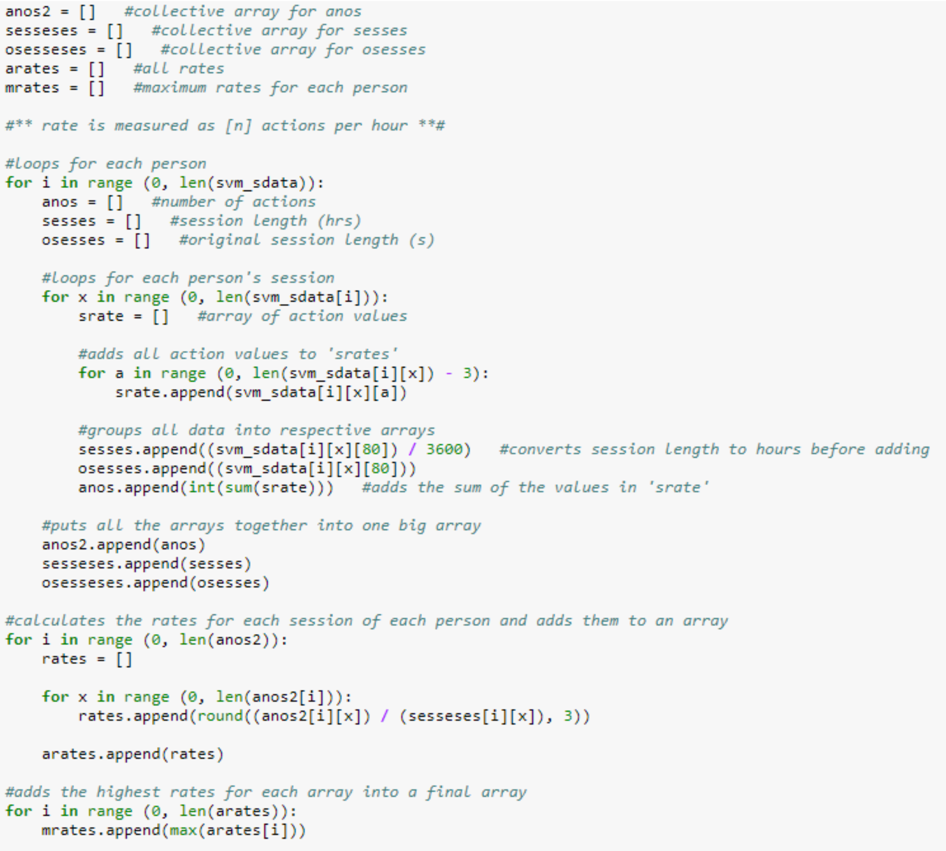




```
Suspicious Time Usernames and Times (hh:mm:ss):
3: [redacted]
34: [redacted]
114: [redacted]
114: [redacted]
137: [redacted]
137: [redacted]
173: [redacted]
```

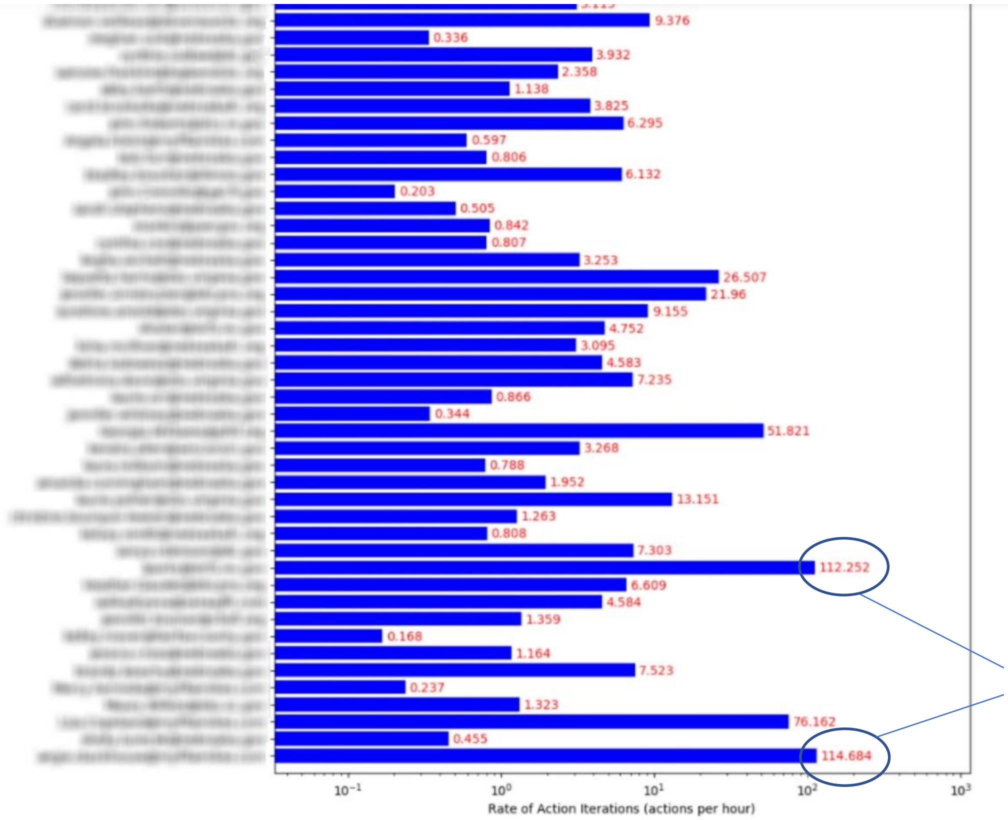
This is what the suspicious logins chart looks like.

The second visualization I programmed was a chart that displayed the iteration rates of each action. More specifically, this chart was a horizontal bar graph that calculated how many times an action, on average, was completed by each user in one hour. Then, only the highest rate for each person is displayed on the graph, so there is only one bar per person and because that is the most important number, because we want to know if there is any user who is completing an action at an alarmingly high rate, e.g. 1000 actions per hour. This can help a human judge, for example, if a user is hoarding data using a bot, or something of the nature. Additionally, implemented a logarithmic x axis scaling, because there were extremely varying values between people, and a normally scaled axis would make it hard to understand the data.



```
anos2 = []   #collective array for anos
sesseses = []   #collective array for sesses
osesseses = []   #collective array for osesses
arates = []   #all rates
mrates = []   #maximum rates for each person

#** rate is measured as [n] actions per hour **#

#loops for each person
for i in range (0, len(svm_sdata)):
    anos = []   #number of actions
    sesses = []   #session length (hrs)
    osesses = []   #original session length (s)

    #loops for each person's session
    for x in range (0, len(svm_sdata[i])):
        srate = []   #array of action values

        #adds all action values to 'srates'
        for a in range (0, len(svm_sdata[i][x]) - 3):
            srate.append(svm_sdata[i][x][a])

        #groups all data into respective arrays
        sesses.append((svm_sdata[i][x][80]) / 3600)   #converts session length to hours before adding
        osesses.append((svm_sdata[i][x][80]))
        anos.append(int(sum(srate)))   #adds the sum of the values in 'srate'

    #puts all the arrays together into one big array
    anos2.append(anos)
    sesseses.append(sesses)
    osesseses.append(osesses)

#calculates the rates for each session of each person and adds them to an array
for i in range (0, len(anos2)):
    rates = []

    for x in range (0, len(anos2[i])):
        rates.append(round((anos2[i][x]) / (sesseses[i][x]), 3))

    arates.append(rates)

#adds the highest rates for each array into a final array
for i in range (0, len(arates)):
    mrates.append(max(arates[i]))
```

This excerpt of code is the main part of the program; this is where I collect all the session lengths and action iterations, convert them into actions per hour, and seek out the highest rate for each user.



Suspicious data

This is a portion of the unusual activity chart.

Conclusions

Being a part of this project, I learned several things. Perhaps one of the most important things I learned was teamwork. Even though I was working with just one colleague for the most part, I learned how to plan together and divide work efficiently and effectively. Another thing I learned was how Python works; I got to familiarize myself with the quirks and features of yet another programming language. Last but definitely not least, I got to experience how a real-life work environment works, which is very different from what I have experienced in school.

I am very confident that this experience will be very helpful in my all my future endeavors because not many high-school students get an opportunity to work on projects like this. I want to thank Mr. Chandra Jonelagadda and Mr. Mike Giammanco of Kovid for providing this opportunity to me, as well as Aneesh Jonelagadda for mentoring me through the whole process.