DARPA Intrusion Detection Dataset Visualization

Alexandra Hurst

The data for this project is from the 1998 DARPA Intrusion Detection Dataset. The dataset is set to mimic real internet traffic across a network. The dataset was fairly clean when downloaded, since it is a high profile and regularly used dataset. To enhance it, I converted the dates and times to a datetime index and turned the 'duration of the connection' column into a timedelta object. Columns were also renamed to better describe what each column represented.

Please note only HTTP/HTTPS connections include data in the 'port' column, since many of the connection types aren't TCP connections and thus don't require a port to connect through.

Also note data is only collected from approximately 8:00 am to 6:00 pm, so large chunks of missing data outside that range is normal.

The purpose of this project is not only to look for malicious traffic, but to demonstrate data visualization techniques that could make finding outliers from any network traffic easier.

In [190]:
```
import numpy as np
import pandas as pd
import datetime
from matplotlib import pyplot as plt
import networkx as nx
%matplotlib inline
plt.rcParams['figure.figsize'] = (8,9)
```

In [168]:
```
days=['monday','tuesday','wednesday','thursday','friday']

traffic=pd.read_csv('training_week_one/monday/tcpdump.list', delim_white
space=True, header=None)

for i in days:
    x=pd.read_csv("training_week_one/"+i+"/tcpdump.list", delim_whitespa
ce=True, header=None)
    traffic=traffic.append(x)
traffic=traffic[[1,2,3,4,6,7,8]]
traffic=traffic.rename(columns={1:'date',2:'time',3:'duration',4:'type',
6:'port',7:'src_ip',8:'dest_ip'})
traffic['date']=pd.to_datetime(traffic['date']+" "+traffic['time'])
traffic['duration']=pd.to_timedelta(traffic['duration'])
traffic=traffic.set_index('date')
traffic=traffic[['duration','type','port','src_ip','dest_ip']]
```

In [181]:
```
traffic['duration_as_int']= traffic['duration'].dt.seconds
print("\t\t\tPreview of the Data")
traffic[5:20]
```

Preview of the Data

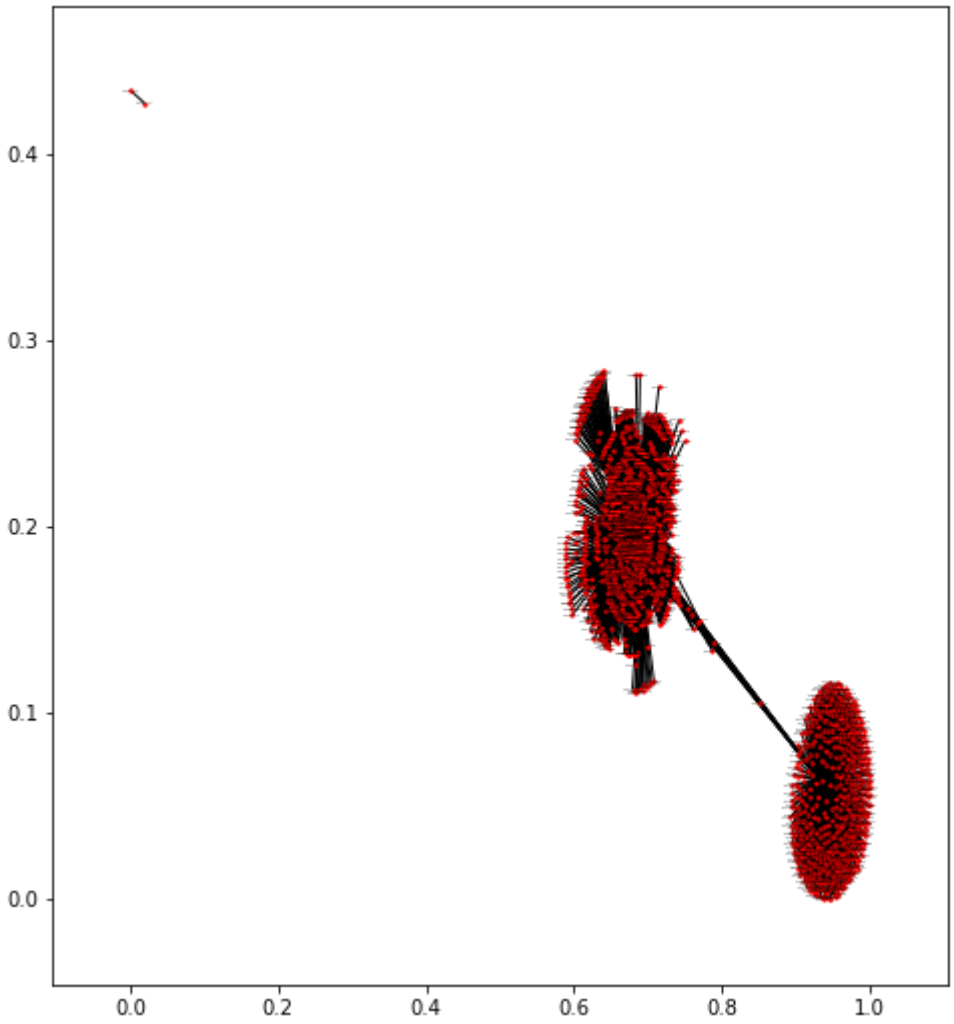
Out[181]:

	duration	type	port	src_ip	dest_ip	duration_as_int
date						
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.214.011	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.214.011	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.214.011	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.212.172	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.212.172	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.214.011	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.214.011	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.212.172	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.214.011	1
1998-06-02 00:00:07	00:00:01	http	80	172.016.114.207	152.163.214.011	1
1998-06-02 00:00:59	00:00:01	ntp/u	123	172.016.112.020	192.168.001.010	1
1998-06-02 00:01:01	00:00:01	eco/i	-	192.168.001.005	192.168.001.001	1
1998-06-02 00:01:21	00:00:01	http	80	172.016.114.207	207.077.090.015	1
1998-06-02 00:01:22	00:00:01	http	80	172.016.114.207	207.077.090.013	1
1998-06-02 00:02:32	00:00:01	http	80	172.016.114.207	152.163.214.011	1

A Network Map of the Data

In [182]:
```
G = nx.from_pandas_dataframe(traffic, 'src_ip', 'dest_ip',['type'])
```

In [191]:
```
nx.draw_networkx(G, layout = nx.shell_layout, node_size=2, font_size = 1
)
plt.show()
```



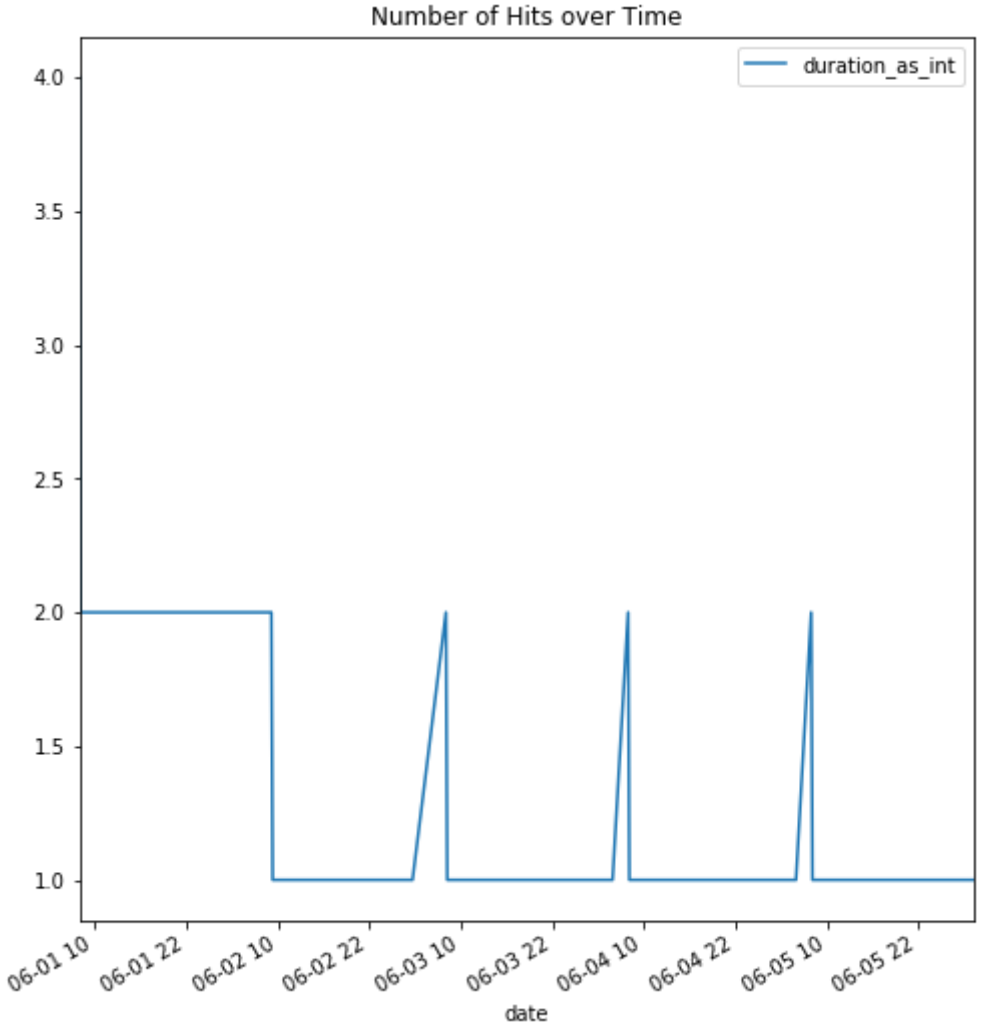
The graph gives us a general picture of the network. There are two seperate halves in the graph that we will look into more.

In [184]:
```
A, B = nx.connected_components(G)
```

A is the larger component. B is the small blip to the side made up of 2 IPs. I will analyze B first.

In [192]:
```
print("B: ", B)
smaller_subset = traffic[traffic['src_ip'].isin(B)]
small_traf = smaller_subset.groupby("date").count()
small_traf[['duration_as_int']].plot(title = "Number of Hits over Time")
plt.show()
```

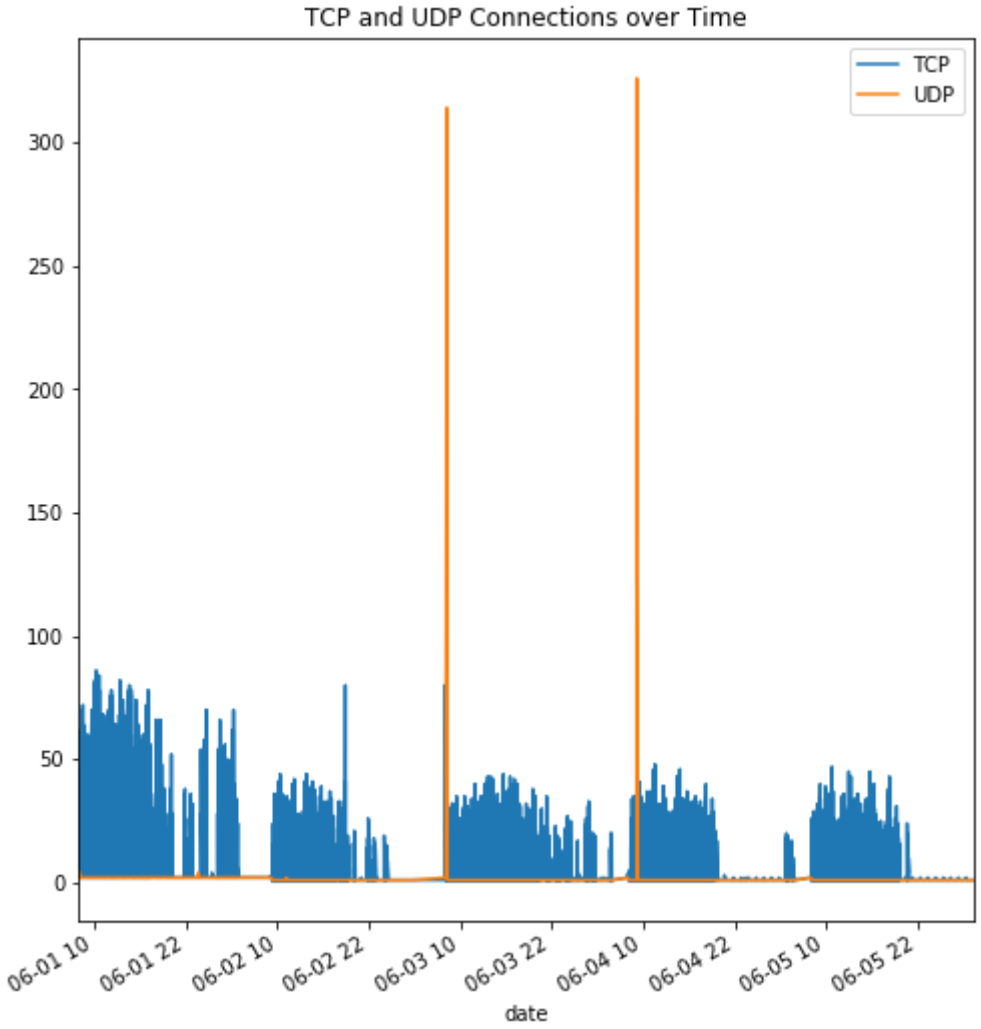
```
B:  {'192.168.001.005', '192.168.001.001'}
```



Since the 2 IPs aren't reaching out to any other IPs and since the traffic fluxuates at such regular intervals, it is possible this is an attack. However, since the IPs both come from the same location, it is more likely that the 2 IPs are part of a higher-security, less-used region of the network. Graphing number of hits over time does allow the user to get a better picture of what's going on over time. Many attacks are carried over regular intervals similar to what is in the graph above.

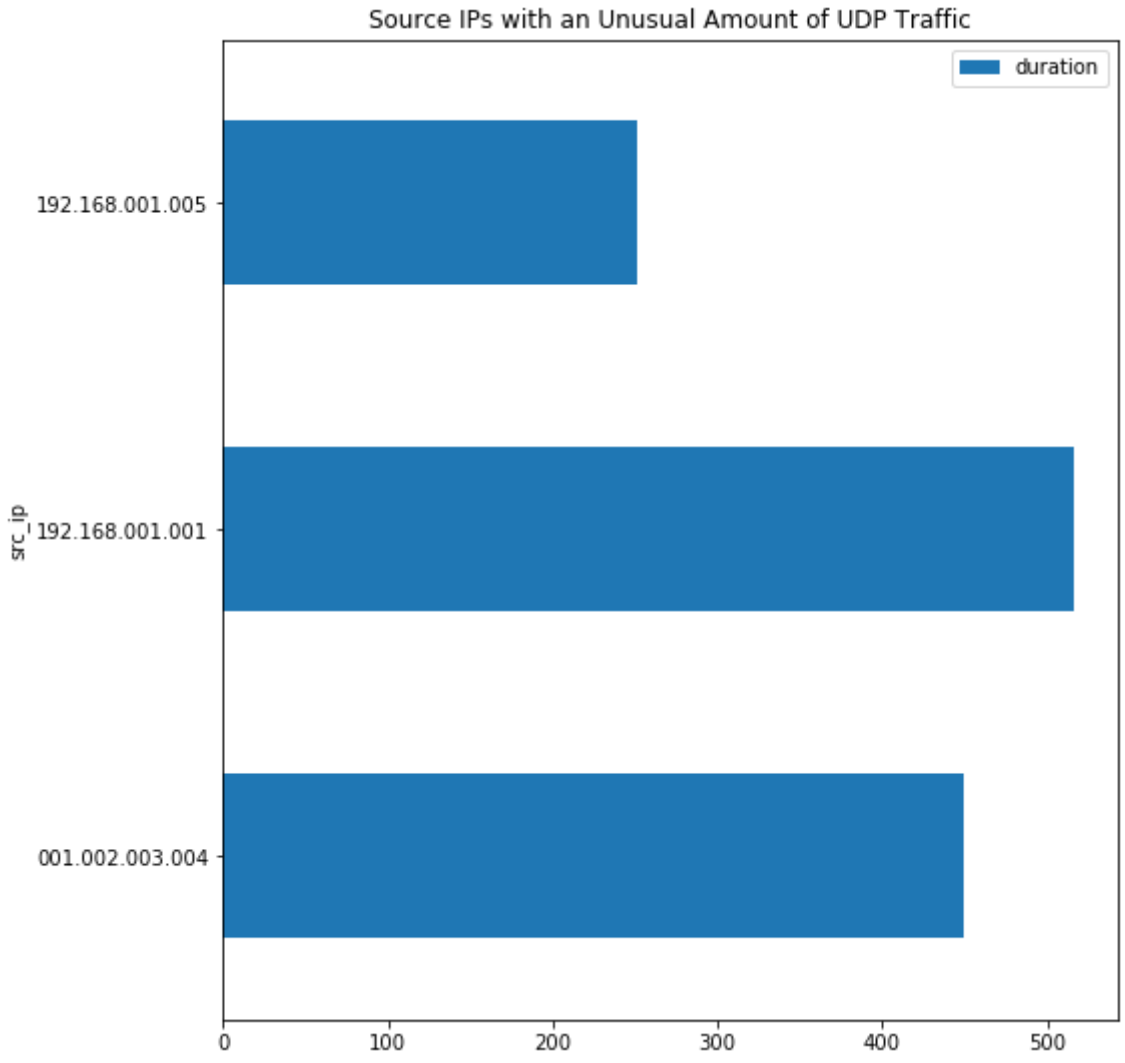
We now analyze A: the larger piece of the graph.

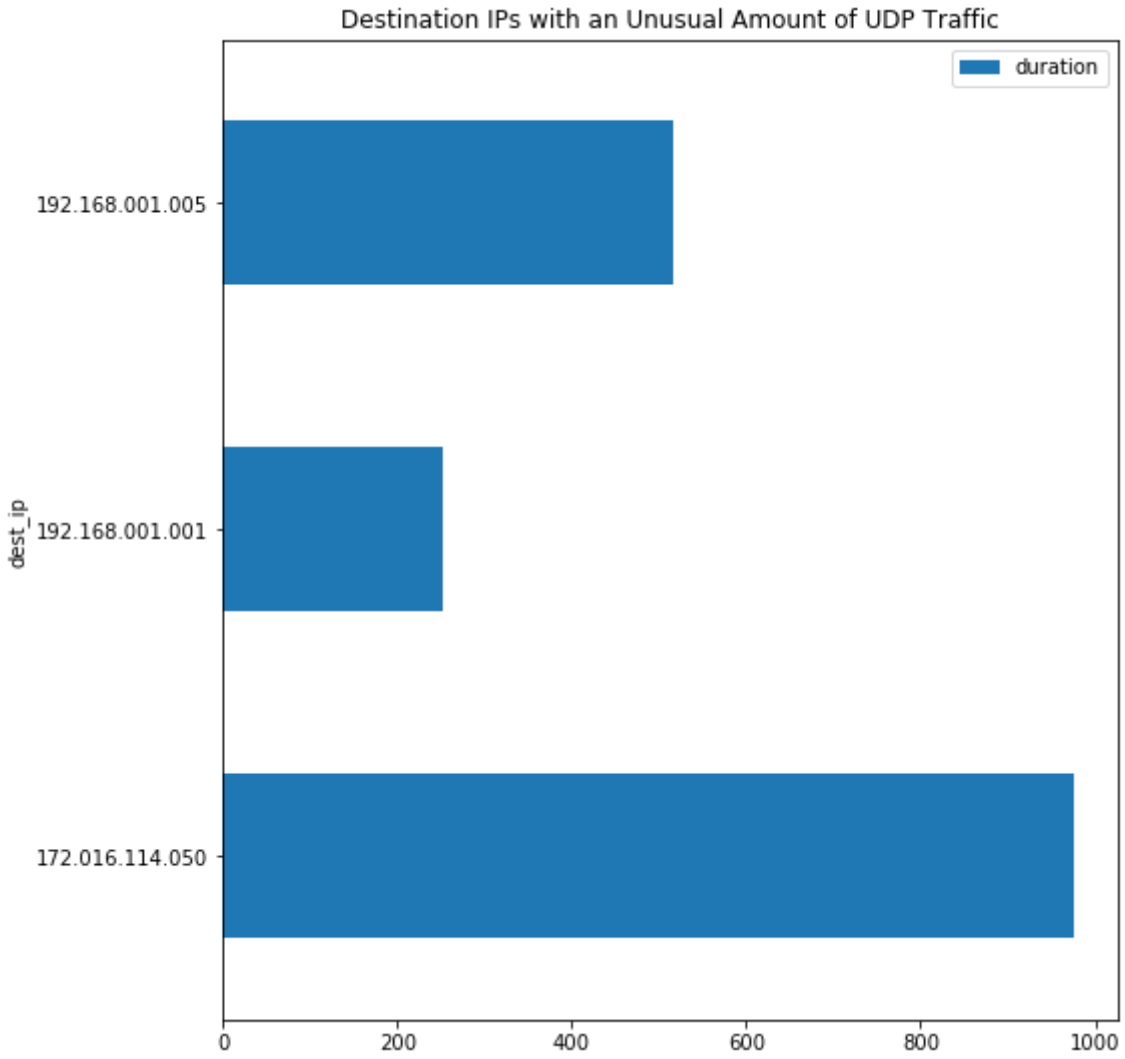
In [193]:
```
larger_subset = traffic[traffic['src_ip'].isin(A)]
http_traf = traffic[traffic['port']!= '-']
other_traf = traffic[traffic['port']== '-']
http_group = http_traf.groupby("date").count()
other_group = other_traf.groupby("date").count()
http_group['duration_as_int'].plot(label = "TCP", title = "TCP and UDP C
onnections over Time")
other_group['duration_as_int'].plot(label = "UDP")
plt.legend()
plt.show()
```



TCP connections are those that require ports (like HTTP/HTTPS). UDP are those that don't. We expect the ratio of TCP to UDP to remain fairly consistent throughout the day. The large orange spikes of UDP traffic should raise some red flags. It could be someone trying to map the network using pings, someone trying to execute a Denial of Service attack, etc. We will now investigate the orange spikes.

In [195]:
```
traffic[traffic['port']=='-']
source_ips = other_traf.groupby("src_ip").count()
source_ips = source_ips[source_ips['duration']>10][['duration']]
source_ips.plot(kind='barh', title= "Source IPs with an Unusual Amount o
f UDP Traffic")
plt.show()
dest_ips = other_traf.groupby("dest_ip").count()
dest_ips = dest_ips[dest_ips['duration']>10][['duration']]
dest_ips.plot(kind='barh', title= "Destination IPs with an Unusual Amoun
t of UDP Traffic")
plt.show()
```


Source IPs with an Unusual Amount of UDP Traffic
duration
src_ip
192.168.001.005
192.168.001.001
001.002.003.004
0
100
200
300
400
500



Those with '192.xxx.xxx.xxx' IPs are from within the network, so those without that beginning have a higher chance of being malicious. In this case, we would analyze '172.016.114.050' and '001.002.003.004'.

Conclusion

Data visualization in cybersecurity can be regularly used identify statistical outliers in the datasets. It generates a smaller list of IPs to be manually inspected than you would otherwise have to look for, thus streamlining the process of finding outliers. At this point, the results of the data visualization and the suspected malicious IPs would be manually analyzed by the Tier 1 Security Analysts in the company.

In [ ]: