SRT411 Assignment 2

Phil King

April 15, 2017

## Introduction to the 6 step Information visualization process

Step 1: Define the Problem. The problem should be use-case driven, not data driven. Answer, what do you need to understand? What do you expect to see? What would you like to see?

Step 2: Assess available Data. What type of data do you need to answer the aforementioned questions? What pieces of data do you need?

Step 3: Process information. Format the data into a usable format for graphical generating resources to work with. If you have ambiguous fields add meta data to the fields which explain the meaning of it. Then parse the data into a database in a usable format. Furthermore you may need to add additional data, fields, or other information from outside the log file, such as geographical location. You may also want to cut out pieces of information that you don’t need. Aggregation: it involves the summarization of data points in the interest of simplifying things so that patterns in the data can be better seen in the graph. For example for some graphs you show an average over time, and sometimes you group data by a collection of ports instead of each port. A common problem with csv data is if your data points have a varied number of roles, it is difficult to collect the machines by roles. A solution to this is only using the most important, or primary, role but then some information gets lost.

Step 4: Visual Transformation. Once you have obtained all the information you want in a CSV file the next step is mapping the data into some visual structure that produces a graphic representation, ie a scatter plot. Here you should choose a graph which can best represent your data. The primary dimension is the findings which you want to convey, you’d like a graph to demonstrate. The size and shape of the graphs are important, you want it to be as small as possible while being legible and easy to understand. Color can be used for two purposes. First it can be used to differentiate various parts of a graph. For example if one node in a graph is red and the rest are green the red node should completely different information. Use the first color according to function, and then the second should contrast it to highlight specific data points.

Step 5: View Transformation. Usually at this point, the graph is not concise, at this point we end up with too many data points. Commonly information is lost through filtering, but sometimes information can be summarized instead of lost.

Step 6: Interpret and Decide. By now wit View transformation you’ve created multiple graphs on the relevant data. Now you ought to be closing in one final graph that satisfies our initial objectives. If you’ve obtained the final graph, all you must do is read it.

## Beginning the process

### Step 1:

Whats the difference between TOR browsing and regular browsing in terms of port usage.  
###Step 2: I need samples of a TOR browsing session and a HTML browsing session. Specifically I need to analyze a difference in protocols used.  
I ran wire shark while browsing the web as both TOR and the regular browser, while tracking the ports ###Step3: I collected some wireshark data during a browsing session on the regular browser and saved it as “Regbrowsing.pcapng”. I then ran the below tshark command to get a csv full of the protocols used.

tshark -r Regbrowsing.pcapng -T fields -e ip.src -e ip.dst -e \_ws.col.Protocol -E header=y -E separator=,> regbrowsing.csv

I then used the below python script to separate the sent packets protocols from the received packets protocols.

#!/usr/bin/python3.5 newfile=open("srcsentreg", "w") newfile.write("protocol\n") newfile2=open("dstsentreg","w") newfile2.write("protocol\n") with open("/home/trump/Documents/SRT411Ass2/regbrowsing.csv") as f: for line in f: if line.split(",")[0]=="192.168.0.19" and line.split(",")[1] != "" and line.split(",")[2] !="\n": newfile.write(line.split(",")[2]) print("1")  
else: if line.split(",")[1]=="192.168.0.19" and line.split(",")[0]!="" and line.split(",")[2]!="\n": print("2") newfile2.write(line.split(",")[2]) ----

I collected some wireshark data during a browsing session on TOR and saved it as “Torbrowsing.pcapng” I then ran the below tshark command to get a csv full of the protocols used.

tshark -r Torbrowsing.pcapng -T fields -e ip.src -e ip.dst -e \_ws.col.Protocol -E header=y -E separator=,> torbrowsing.csv

## I then used the below python script to separate the sent packets protocols from the received packetse protocols.

#!/usr/bin/python3.5 newfile=open("srcsenttor", "w") newfile.write("protocol\n") newfile2=open("dstsenttor","w") newfile2.write("protocol\n") with open("/home/trump/Documents/SRT411Ass2/torbrowsing.csv") as f: for line in f: if line.split(",")[0]=="192.168.0.19" and line.split(",")[1] != "" and line.split(",")[2] !="\n": newfile.write(line.split(",")[2]) print("1")  
else: if line.split(",")[1]=="192.168.0.19" and line.split(",")[0]!="" and line.split(",")[2]!="\n": print("2") newfile2.write(line.split(",")[2]) -----

I then brought the produced files into R, and used the table function to summarize them. The variables start off with the browser kind (t/r) for TOR/regular, followed by the direciton (s/r) for sent/recieved. Making tor sent becoming "ts". I then run table to summarize them

ts<- read.csv2("/home/trump/Documents/SRT411Ass2/dstsenttor")  
table(ts)

## ts  
## SSL SSLv2 TCP TLSv1.2 UDP   
## 33 1185 9628 22 4

tr<- read.csv2("/home/trump/Documents/SRT411Ass2/srcsenttor")  
table(tr)

## tr  
## IGMPv3 LLMNR NBNS SSDP TCP TLSv1.2 UDP   
## 2 4 3 2 3187 7 11

rr<- read.csv2("/home/trump/Documents/SRT411Ass2/srcsentreg")  
table(rr)

## rr  
## IGMPv3 SSDP TCP TLSv1.2 UDP   
## 3 4 1293 3 5

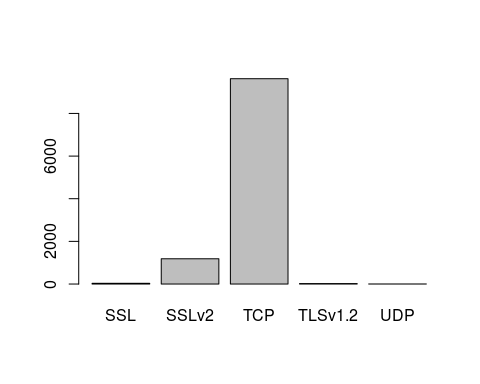
rs<- read.csv2("/home/trump/Documents/SRT411Ass2/dstsentreg")  
table(rs)

## rs  
## TCP TLSv1.2 UDP   
## 5560 540 2

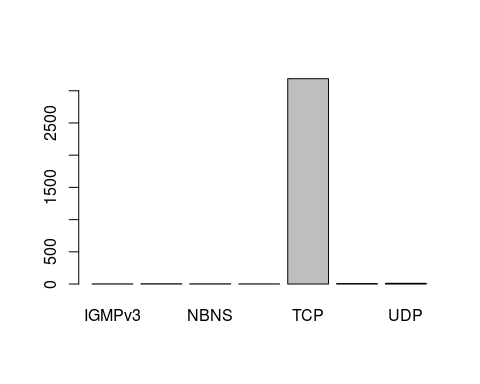
### Step 4:

I then to start made a bar graph for each of them to represent the proportions of each protocol.

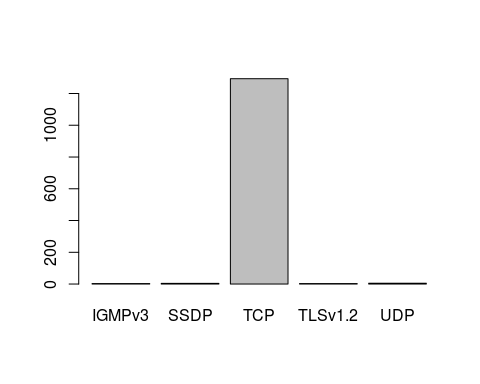
barplot(table(ts))



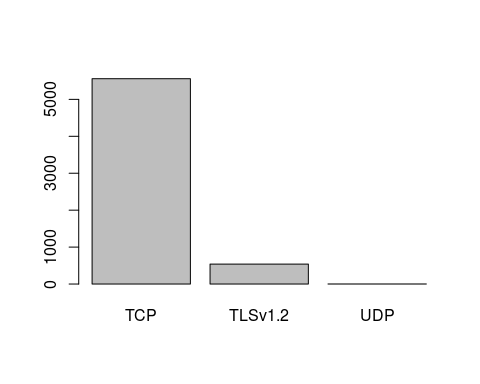
barplot(table(tr))



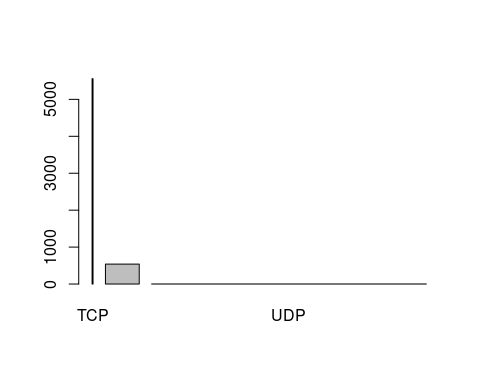
barplot(table(rr))



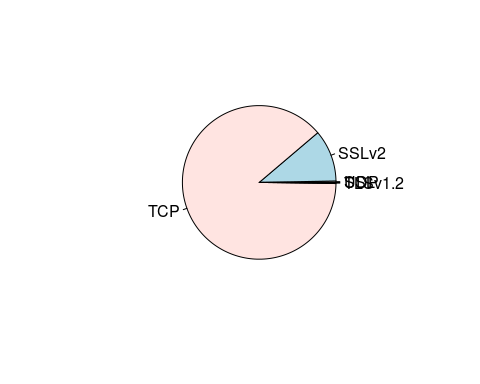
barplot(table(rs))

 I then tried to make a comparative barplot using barplot(table(rs),table(ts)), but this proved not useful as the samples sizes were not equal in size.

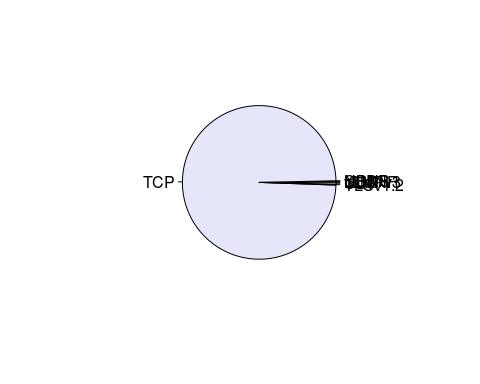
barplot(table(rs),table(ts))

 I must convert the raw numbers to percentages, or use a different graph. I then tried Pie graphs. The results proved hard to read in some cases, and the data still does not compare against each other.

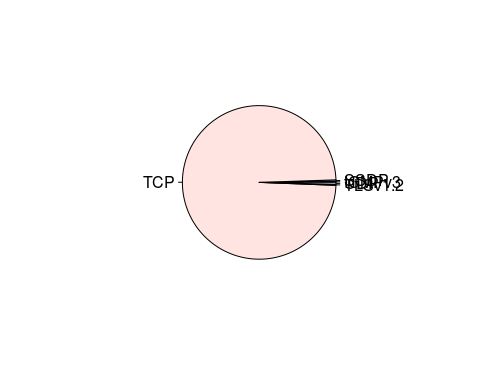
pie(table(ts))



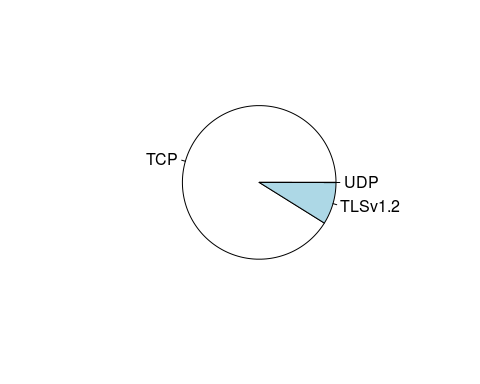
pie(table(tr))



pie(table(rr))



pie(table(rs))

 I then used cbind to convert the counts into percentages.

tbl<-table(ts)  
tss<-cbind(tbl,prop.table(tbl))  
tss

## tbl   
## SSL 33 0.0030353201  
## SSLv2 1185 0.1089955850  
## TCP 9628 0.8855776306  
## TLSv1.2 22 0.0020235467  
## UDP 4 0.0003679176

tbl<-table(tr)  
trs<-cbind(tbl,prop.table(tbl))  
trs

## tbl   
## IGMPv3 2 0.0006218905  
## LLMNR 4 0.0012437811  
## NBNS 3 0.0009328358  
## SSDP 2 0.0006218905  
## TCP 3187 0.9909825871  
## TLSv1.2 7 0.0021766169  
## UDP 11 0.0034203980

tbl<-table(rr)  
rrs<-cbind(tbl,prop.table(tbl))  
rrs

## tbl   
## IGMPv3 3 0.002293578  
## SSDP 4 0.003058104  
## TCP 1293 0.988532110  
## TLSv1.2 3 0.002293578  
## UDP 5 0.003822630

tbl<-table(rs)  
rss<-cbind(tbl,prop.table(tbl))  
rss

## tbl   
## TCP 5560 0.9111766634  
## TLSv1.2 540 0.0884955752  
## UDP 2 0.0003277614

### Step 5:

While trying to make a comparative barplot I noticed the number of rows didn't match, so I manually added rows with 0's and the missing protocol. I then reloaded them, and created 2 new lists. The new lists now have the sender tor/regular table data in one table and the reciever tor/regular table data in another table.

library(plotly)

## Loading required package: ggplot2

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following object is masked from 'package:stats':  
##   
## filter

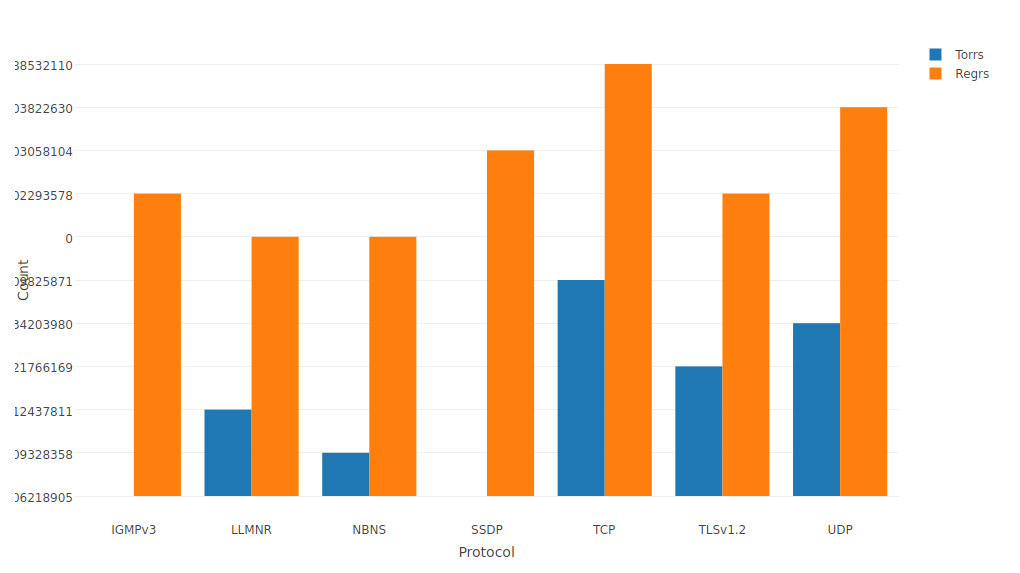
## The following object is masked from 'package:graphics':  
##   
## layout

rrs<-read.csv2("/home/trump/Documents/SRT411Ass2/rrs",sep=",")  
trs<-read.csv2("/home/trump/Documents/SRT411Ass2/trs",sep=",")  
rss<-read.csv2("/home/trump/Documents/SRT411Ass2/rss",sep=",")  
tss<-read.csv2("/home/trump/Documents/SRT411Ass2/tss",sep=",")  
trrs<-mutate(trs,rrs=rrs[,3])  
trss<-mutate(tss,rss=rss[,3])

### Step 6:

Now I have everything prepared to make the comparative bargraph. Below is the percentage comparison of protocols seen in the recieved packets of the TOR and regular browser data. What is witnessed is the different distribution of the protocols.

Protocol<-trrs$protocol  
Torrs<-trrs$perc  
Regrs<-trrs$rrs  
plot\_ly(trrs, x = ~Protocol, y = ~Torrs, type = 'bar', name = 'Torrs') %>%  
 add\_trace(y = ~Regrs, name = 'Regrs') %>%  
 layout(yaxis = list(title = 'Count'), barmode = 'group')



Below is the percentage comparison of protocols seen in the sent packets of the TOR and regular browser data

Protocol<-trss$protocol  
Torss<-trss$perc  
Regss<-trss$rss  
  
  
plot\_ly(trss, x = ~Protocol, y = ~Torss, type = 'bar', name = 'Torss') %>%  
 add\_trace(y = ~Regss, name = 'Regss') %>%  
 layout(yaxis = list(title = 'Count'), barmode = 'group')

