MAKING PRETTY GRAPHS - RRDTOOL

What does making pretty graphs have to do with security or even system administration? Well, if they help you to see at a glance what is going on, then a lot (it also helps that pretty graphs are the only document that management understands). So lets take a look at what you need to get these pretty graphs. We are going to be using *rrdtool*. We will be using this tool because it can make graphics which can be used in many ways, and the database never really grows (*rrd* stands for round-robin-database) and lastly, anything you can put a number to, it can graph.

Getting rrdtool
You do of course need linux. You will also need a couple of other bits and pieces (please bear in mind that these were the versions when I did it, check for newer ones when you try)..

```
wget http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/pub/rrdtool-1.2.11.tar.gz
wget http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/pub/libs/cgilib-0.5.tar.gz
wget http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/pub/libs/libart_lgpl-2.3.17.tar.gz
wget http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/pub/libs/libpng-1.2.8-config.tar.gz
```

Then you need to do the install *cgilib, libart, libpng* then *rrdtool* doing the normal *./configure , make , make install* dance (you may need to set *CPPFLAG* variable for *rrdtool*). Assuming thats all gone well, let move on to creating your first database.

Creating your first rrd database
For my example database I am going to try to use a topical example. Lets graph your outgoing traffic. We want to create a graph showing different lines for different protocols. This will be displayed via a webpage. Sounds cool? Well it all starts with the database. Lets create a */admin/graph* directory first, and in there we have a *create_out_rrd* script (the line numbers I added, remove them for a working script)..

```
1:rrdtool create /admin/graph/cntout.rrd --start N --step 120 \
2:DS:http_out:GAUGE:600:U:U \
3:DS:https_out:GAUGE:600:U:U \
4:DS:smtp_out:GAUGE:600:U:U \
5:DS:pop_out:GAUGE:600:U:U \
6:DS:msn_out:GAUGE:600:U:U \
7:DS:ssh_out:GAUGE:600:U:U \
8:DS:ftp_out:GAUGE:600:U:U \
9:DS:dns_out:GAUGE:600:U:U \
10:DS:unknown_out:GAUGE:600:U:U \
11:RRA:AVERAGE:0.5:1:600 \
12:RRA:AVERAGE:0.5:6:700 \
13:RRA:AVERAGE:0.5:24:775 \
14:RRA:AVERAGE:0.5:288:797 \
15:RRA:MAX:0.5:1:600 \
16:RRA:MAX:0.5:6:700 \
17:RRA:MAX:0.5:24:775 \
18:RRA:MAX:0.5:288:797
```

Lets see what all that actually does..

Line	*Explanation*
1	This issues the actual create option, gives the name (*/admin/graph/cntout.rrd*), tells the time to start now (*--start N*) and finally tells the database that it will be updated every 120 seconds, also known as every 2 minutes (*--step 120*)
2-10	These are the "data stores" (*DS*). Think of these as the fields which store the value each time you update the database. The second field is a unique name. The third field is the type of field used, I am using *GAUGE* because I will be using specific numbers each time I update the database. If your counter does not zero after each read, then use *COUNTER* instead, The other options are *ABSOLUTE* and *DERIVATIVE*. Next we say that each field will use the values as average out over 600 seconds. The last two fields basically mean we do not put any data or time limits on our fields.

11-14	These are basically entries that allow us to store average values for each of our data stores
15-18	Same as above except we keep the maximum values for our data stores

Once you run the script, you should see the */admin/graph/cntout.rrd* file appear. Congratulations! You have created your first rrd database. Now lets populate it.

Putting the data in

Ok, remember when you created your database? you told it you were going to update it every two minutes. So you are going to need a script that you can put into your *crontab* to run every two minutes. Since you are going to be querying your firewall for outgoing statistics, you need to make sure that your firewall can give them to you. I will assume that you have an *iptables* firewall (anything else and you need to figure out how to get the numbers on your own), you can use these rules to create a suitable chain..

```
echo "--TRAFFIC ACCOUNTING - OUT"
$IPT -N CNTOUT
for TCPAMTOUT in 20 80 22 21 25 110 443 1863
 do
  $IPT -A CNTOUT -o $EXT -p tcp --dport $TCPAMTOUT -j RETURN
 done
for UDPAMTOUT in 53
 do
  $IPT -A CNTOUT -o $EXT -p udp --dport $UDPAMTOUT -j RETURN
 done
$IPT -A CNTOUT -j RETURN
$IPT -A FORWARD -o $EXT -j CNTOUT
```

Now you need to get your server with *rrdtool* on it, to query this chain. Lets create the */admin/graph/update_out* script (the line numbers I added, remove them for a working script)..

```
1:TMP=/tmp/cnt.out.kp
2:/usr/bin/ssh 192.168.12.13 "iptables -vnxL CNTOUT -Z" > $TMP
3:
4:HTTP=`cat $TMP | grep -e "dpt:80" | gawk '{ print $2 }'`
5:HTTPS=`cat $TMP | grep -e "dpt:443" | gawk '{ print $2 }'`
6:SMTP=`cat $TMP | grep -e "dpt:25" | gawk '{ print $2 }'`
7:POP=`cat $TMP | grep -e "dpt:110" | gawk '{ print $2 }'`
8:MSN=`cat $TMP | grep -e "dpt:1863" | gawk '{ print $2 }'`
9:SSH=`cat $TMP | grep -e "dpt:22" | gawk '{ print $2 }'`
10:FTP=`cat $TMP | grep -e "dpt:21" | gawk '{ print $2 }'`
11:FTPDATA=`cat $TMP | grep -e "dpt:20" | gawk '{ print $2 }'`
12:FTP=`expr $FTP \+ $FTPDATA`
13:DNS=`cat $TMP | grep -e "dpt:53" | gawk '{ print $2 }'`
14:UNKNOWN=`cat $TMP | grep -e "RETURN" | grep -v "dpt:" | gawk '{ print $2 }'`
15:
16:HTTPKP=`expr $HTTP \/ 1024`
17:HTTPSKP=`expr $HTTPS \/ 1024`
18:SMTPKP=`expr $SMTP \/ 1024`
19:POPKP=`expr $POP \/ 1024`
20:MSNKP=`expr $MSN \/ 1024`
21:SSHKP=`expr $SSH \/ 1024`
22:FTPKP=`expr $FTP \/ 1024`
23:DNSKP=`expr $DNS \/ 1024`
24:UNKNOWNKP=`expr $UNKNOWN \/ 1024`
25:
26:/usr/local/bin/rrdtool updatev /admin/graph/cntout.rrd N:\
27:$HTTPKP:$HTTPSKP:$SMTPKP:$POPKP:$MSNKP:$SSHKP:$FTPKP:$DNSKP:$UNKNOWNKP
28:
29:rm -rf $TMP
30:/admin/graph/graph_out_hourly
31:/admin/graph/graph_out_daily
32:/admin/graph/graph_out_weekly
```

Lets step through this script to see what it does..

Line	*Explanation*

1	Setting a variable
2	This is where you query the firewall chain you created. This command assumes you can login to the firewall via ssh without being asked for a password. The iptables options will give you the table listing and then zero all the counters
4-14	Here we go through the table listing for each protocol and assign each variable the number of bytes (the second column) which the firewall has seen go through it. The exception here is the ftp data, here we first get the values for port 21 and then 20, we add them and that sum is assigned to the variable.
16-24	Here each variable is divided by 1024 to give you kilobytes as the way we queried the firewall gives us the traffic size in bytes
26-27	This is where we use the rrdtool binary to update the database. You can see we have specified the database to update, and then we give each variable in the order we created the fields in the intital database
29	We remove our temporary file
30-32	We create our graphs after each update. You can see we are creating a graph for hourly, daily and weekly stats. More on that later

Now before we actually run or schedule this, lets create the graphing scripts.

Putting the data into pictures
In the graph script we will look at shortly, I have used a couple of non-basic options because I think they actually make the end result better, please feel free to tweak. So lets create our */admin/graph/graph_out_hourly* (again the line numbers I put in must be removed for a working script)..

```
1:/usr/local/bin/rrdtool graph /var/www/html/graph/outhrly.png -t "Out-of-Company Hourly Bandwidth" --start
-3600 \
2: -c MGRID#E0E1E5 --no-minor -w 600 -h 200 -X 0 -A --vertical-label kilobytes \
3: DEF:https=/admin/graph/cntout.rrd:https_out:AVERAGE \
4: DEF:http=/admin/graph/cntout.rrd:http_out:AVERAGE \
5: DEF:smtp=/admin/graph/cntout.rrd:smtp_out:AVERAGE \
6: DEF:pop=/admin/graph/cntout.rrd:pop_out:AVERAGE \
7: DEF:msn=/admin/graph/cntout.rrd:msn_out:AVERAGE \
8: DEF:ssh=/admin/graph/cntout.rrd:ssh_out:AVERAGE \
9: DEF:ftp=/admin/graph/cntout.rrd:ftp_out:AVERAGE \
10: DEF:dns=/admin/graph/cntout.rrd:dns_out:AVERAGE \
11: DEF:unknown=/admin/graph/cntout.rrd:unknown_out:AVERAGE \
12: LINE1:http#00FF00:"HTTP" \
13: LINE1:https#CEA5E4:"HTTPS" \
14: LINE1:smtp#E43348:"SMTP" \
15: LINE1:pop#8B1AE4:"POP" \
16: LINE1:msn#C0C0C0:"MSN" \
17: LINE1:ssh#E4A979:"SSH" \
18: LINE1:ftp#9BE4E4:"FTP" \
19: LINE1:dns#125FE4:"DNS" \
20: LINE1:unknown#1E1B1D:"UNKNOWN" \
21: GPRINT:http:LAST:'LAST RATE HTTP\: %5.0lf kilobytes ' \
22: GPRINT:http:MAX:'MAX RATE HTTP\: %5.0lf kilobytes \j' \
23: GPRINT:https:LAST:'LAST RATE HTTPS\: %5.0lf kilobytes ' \
24: GPRINT:https:MAX:'MAX RATE HTTPS\: %5.0lf kilobytes \j' \
25: GPRINT:smtp:LAST:'LAST RATE SMTP\: %5.0lf kilobytes ' \
26: GPRINT:smtp:MAX:'MAX RATE SMTP\: %5.0lf kilobytes \j' \
27: GPRINT:pop:LAST:'LAST RATE POP\: %5.0lf kilobytes ' \
28: GPRINT:pop:MAX:'MAX RATE POP\: %5.0lf kilobytes \j' \
29: GPRINT:msn:LAST:'LAST RATE MSN\: %5.0lf kilobytes ' \
30: GPRINT:msn:MAX:'MAX RATE MSN\: %5.0lf kilobytes \j' \
31: GPRINT:ssh:LAST:'LAST RATE SSH\: %5.0lf kilobytes ' \
32: GPRINT:ssh:MAX:'MAX RATE SSH\: %5.0lf kilobytes \j' \
33: GPRINT:ftp:LAST:'LAST RATE FTP\: %5.0lf kilobytes ' \
34: GPRINT:ftp:MAX:'MAX RATE FTP\: %5.0lf kilobytes \j' \
35: GPRINT:dns:LAST:'LAST RATE DNS\: %5.0lf kilobytes ' \
36: GPRINT:dns:MAX:'MAX RATE DNS\: %5.0lf kilobytes \j' \
```

```
37: GPRINT:unknown:LAST:'LAST RATE UNKNOWN\: %5.0lf kilobytes ' \
38: GPRINT:unknown:MAX:'MAX RATE UNKNOWN\: %5.0lf kilobytes \j' \
```

Now before you run away, lets just look at this script in sections..

Line	*Explanation*
1	Here we use *rrdtool* to create the graph with the *graph* keyword. We then specify what file and where to create it (*/var/www/html/graph/dbnouthrly.png*), we give it a title (*-t "Out-of-Company Hourly Bandwidth"*) and lastly he set it to graph for the last 3600 seconds, which is one hour (*--start -3600*). If you want to create a daily script, you can use the exact same script except change the start value to 86400 (24 hrs X 60 min. X 60 secs). If you also want weekly then you can work out the number of seconds in a week and use that.
2	Here we set some non-default options. First I make the major gridlines colour a bit lighter (*-c MGRID#E0E1E5*), I remove the minor gridlines (*--no-minor*), I make my graphs a bit larger then normal (*-w 600 -h 200*), I take off the autoscaling of the Y-axis values (*-X 0*), I make the Y-axis max/min values dependant on the data being graphed (*-A*) and I lastly give my Y-axis a title (*--vertical-label kilobytes*).
3-11	Each line here is a definition (*DEF*), the second field names the definition which is pointed to the relevant *rrd* database, the third field is which database field must be referenced, and lastly the type of value we want (here we use *AVERAGE*). This is done for each line/value we want to graph.
12-20	Each line here defines how the line is drawn. *LINE1* means the line will be 1 pixel thick, you can use *LINE2* for a 2 pixel thick line or *AREA* if you want the area shaded in. The next field references the definition the line is for and the colour the line must be. The third field gives the key/title for that line. Each *DEF* has a *LINE1* statement here.
21-38	Each *GPRINT* line allows us to write a short informative point. The second field states the definition we are going to reference. The third field states the type of value we want out, we are using both *LAST* (the very last value input into the *rrd* database) and *MAX* (the largest value for the definition in the timespan being graphed). The last field is what is actually shown on the graph, you will see that it is written like a normal C string (*/j* = right justify) and the *%5.0lf* means the number will have 5 digits to the left of the decimal and 0 to the right.

Ok, now create a script for daily and weekly graphs to match your update script.

Scheduling and displaying
Now that we have everything, lets schedule our update script in crontab. Use an entry like this..

```
*/2 * * * * /admin/graph/update_out
```

And finally lets create a webpage to view these graphs, lets create a */var/www/html/out.html*...

```
<html>
<head>
<meta http-equiv="refresh" content="300" >
</head>
<body>

<img alt="" src="./graph/outhrly.png"><br><br>
<img alt="" src="./graph/outdly.png"><br><br>
<img alt="" src="./graph/outwkly.png"><br><br>

</body>
</html>
```

Nothing fancy, this page will refresh itself every 300 seconds and display the three graphs. Any headings or such-like you can add.

Final Words

I hope this example helps you get to grips with what rrdtool can do. It is a very powerful, quick graphing tool and helps us keep an eye on things as well as make pretty pictures for managers. Remember, anything you can put a number to, you can graph. Rrdtool also has many more options and tweaks, please explore it, have fun and learn.