Message Triage with Perl

by JP Vossen

In the course of administering and monitoring any computing environment, you will receive many system-generated messages about many events. How do you determine which events are important and which are not, which require a response and which do not? You can often answer these questions from prior experience, general knowledge about your environment and the industry, and perhaps from corporate policy and procedures documents. But what if you are not sure what is happening? You will have to do some analysis to find out.

For the purposes of this discussion an event is an occurrence in the computing environment and a message is a computer generated log or notification about that event. Therefore you will receive messages in the form of IDS alerts, logs from hosts, firewalls, routers, etc. From all these you determine what events are taking place and what your reaction should be. Not every message is as clear cut as we would like, and even for reasonably straightforward messages you often need to examine the devices in question more closely to determine what is really going on. The same problems comes up when considering message tuning, such as for an Intrusion Detection Systems (IDS).

How do you know where to start when there are hundreds or thousands of messages to examine? Scrolling through the first few hundred events may give you a sense of things, but I find that events tend to clump up, and there may be a much larger clump farther down in the file. In order to make sense of the messages and find the underlying event, it's important to have some kind of summary to allow you to focus on getting the most results for

Figure 1

272319 Nmap UDP Port Sweep

IP Pair Cnt	Pair: SIP	Pair: DIP
9,290	10.20.72.97	10.20.2.7
8,978	10.20.72.92	10.20.2.7
8,802	10.20.72.33	10.20.2.7
8,010	10.20.72.19	10.20.2.7
7,982	10.20.72.32	10.20.2.7
7,619	10.20.72.56	10.20.2.7
5,104	10.20.72.71	10.20.2.7
5,085	192.168.240.24	192.168.91.39
4,987	10.20.72.20	10.20.2.7
4,795	192.168.240.24	192.168.91.40
…	…	…

SIP Cnt	SIP
40,741	172.29.152.21
34,819	192.168.240.24
20,564	10.20.2.12
12,989	172.29.152.22
9,290	10.20.72.97
8,978	10.20.72.92
7,210	10.20.72.34
7,076	10.20.72.89
6,454	10.20.72.37
5,500	10.20.72.83
…	…

DIP Cnt	DIP
111,189	10.20.2.7
5,114	192.168.91.39
4,821	192.168.91.40
3,364	172.29.125.107
3,344	172.29.135.253
3,273	10.20.75.58
2,776	172.29.133.12
2,711	10.20.75.63
2,696	172.29.135.207
2,412	172.29.133.75
…	…

your analysis and tuning time. There are a number of ways to look at this kind of data, but the one I've found to be most useful I call the message triage view.

According to my dictionary, triage is "assignment of degrees of urgency" which is an important step in event analysis or message tuning. What devices are involved and what is happening? How important is it? In "event analysis" you must sift through messages to determine what is happening and how to respond. Later on you may need to do "message tuning," which is a more specific process that usually revolves around an IDS. Most IDSs are "noisy" until you tune them for your specific needs and environment. Some of that tuning involves your risk factors, and specific details of your machines and environment, but some of it is simply tracking down known false positive traffic and creating an exception for it.

I firmly believe in the 80/20 principle—20% of the effort achieves 80% of the results, and the remaining 20% of the results requires 80% of the effort. Often, 80% of the results are good enough, especially if multiple passes at the problem are acceptable or useful, which is likely true for message triage.

The Process

Starting with the noisiest device gives you the most bang for your buck. If you can correct or mitigate the root cause, or tune the messages if the event is benign or not important on the noisiest device first, you've gotten 80% of your results with 20% of your effect. Make multiple passes at the new noisiest device until you've finished or reached a point of diminishing returns. This is the fastest and most efficient way to proceed, but it only works if you can find the noisiest device. The message triage process tells you where to start.

It's actually three different views, and the concept is very simple. All three "views" deal with a single message at a time, and refer only to the source and destination IP Addresses. Ports are not considered as they are often not relevant when dealing with host logs, and are usually implied when dealing with IDS events. The first view considers the source and destination addresses as a pair—a single unit—from SOURCE-A to DESTINATION-A, from SOURCE-B to DESTINATION-B, etc. The second view considers only the source address and the third only the destination address. Each view is ordered in a descending numerical sort by number of events, and all are presented adjacent to each other. This is sometimes confusing at first, but it is critical to the way the process works. For example, consider the Cisco Secure IDS (CSIDS) alert in Figure 1 on page 9.

From this simple example, you can very quickly see that there are a lot of pairs of machines in-

Message Analysis with a Spreadsheet

Why

If you spend any time administering or securing computers, you have to read a lot of logs. There are a lot of tools that can help with that, such as the UNIX Text Utilities, Perl scripts, and even databases. But they are not always available and sometimes they are not the best tool for the job. Sometimes you just need to look at a big table of log data. Maybe you don't even know what you're looking for. Or maybe you are on a Windows box at someone else's desk and don't have all your usual tools.

Whatever the reason, a spreadsheet can actually be a pretty good log analysis tool. Nearly every desktop is going to have some kind of spreadsheet package. I'll talk about two of the most common, Microsoft Excel and Openoffice.org Calc, but the concepts (if not features) work across any spreadsheet.

How

The first task is to load the data into the spreadsheet. Be aware that Excel can load only 65,536 rows, and Calc is even more limited at 32,000. Any spreadsheet program will be able to open CSV (comma separated value) files, and you can open flat files (such as syslog, or Snort alert files) almost as easily. Windows Event Logs are a lot trickier, since you must first get the data out of the Event Log, and the Event Viewer offers some of the features of a spreadsheet anyway.

For this article, we will examine a flat syslog file. In Excel, you may have to set "Files of type" to 'All files,'

volved in this traffic, there are four machines that are sending an awful lot of it, but there is one machine that is receiving the vast majority of it. It's as simple as that. You now know where to start. If this is an incident response scenario you had better go check out 10.20.2.7 right away. If you are researching a known false positive and have the ability to enter some kind of exception or filter, you now have the addresses that will give you the most bang for the buck.

The Theory

All of the above pre-supposes that you have some kind of monitoring architecture in place. If you don't, you will never see the messages so this is all a moot point. But if you do have some kind of ability to monitor your environment (and you certainly should) you also know that messages come in an amazing variety of formats. Handling each one individually is a challenge, so instead I've defined a simple standard format that's used by the Perl script that generates the triage views. There is a separate pre-parsing step to convert your logs into this format.

Pre-parser

The pre-parser is another Perl script I wrote a while ago to act as a general framework for doing arbitrary parsing. It's name is simply Parser.pl, and it accepts a configuration file that defines how to read the input, parse it, and write the output. The configuration file is actually Perl code as well, and parser.pl is really just a Perl shell or wrapper around it.

Message Triage

MessageTriage.pl is the main Perl script that accepts the standard format generated by Parser.pl and creates a triage view such as the example above. It reads and writes CSV (comma separated value) formatted files that may be read and written by just about any spreadsheet or database program. I also have a simple general purpose Perl script to convert CSV output to HTML.

When running a triage report periodically, say every week for IDS tuning purposes, it can be helpful to have some state information. What has changed from this week to last week? Since MessageTriage.pl already parses the messages, it is trivial to write a separate file containing just the message name or text for this purpose.

The Procedure

The exact procedure you will need to follow depends on your message source or sources, your immediate goal, and the output you prefer. In general, it's the following. See the command line

then choose and open the file. You will get a dialog box that allows you to parse the data. For syslog, use the 'fixed width' type and place delimiters after the date and time. Unless all your hostnames are the same length, you will not be able to isolate them yet. Work though the rest of the wizard as necessary. For Openoffice.org (OO) Calc, the process is the same, except you may have to lie and copy or rename the log file so it has a .csv extension, otherwise OO will open it in the word processor.

Either way, when you are finished, you should have three columns, date, time and the rest of the message. You should to insert a row and add headers, and use the Windows, Freeze (Panes) command to lock the header at the top. Date, time and other may be good enough, but if it's not we can split it out.

Splitting delimited or fixed width text is easy. You could have done that when you imported, and Excel has a wizard in the Data, Text to Columns menu. Splitting text on an arbitrary boundary, such as isolating the first word (hostname) is harder. You can do it with these formula though. Note that the only differences are that Excel uses a comma separator while Calc uses a semi colon, and Calc's MID command is limited to 9999 remaining characters.

Once you have the formula in the top row, you can easily copy it to all other rows as follows.

	Sample cell "C2"	Cell D2 Formula	Cell E2 Formula
Excel	hostname more data goes here	=LEFT(C2,FIND(" ",C2))	=MID(C2,FIND(" ",C2)+1,999999)
Calc	hostname some more data	=LEFT(C2;FIND(" ";C2))	=MID(C2;FIND(" ";C2)+1);9999)

examples below for some specific things to try.

1. Isolate the messages to examine in a flat ASCII files by type. Use find (Windows) or grep (UNIX) to split different message types (.e.g. firewall and IDS) into different flat files.
2. Pre-parse each file into the CSV file expected by MessageTriage.pl using Parser.pl and the appropriate *.parser configuration file.
3. Process the CSV file using MessageTriage.pl.
4. Open and examine the output.

Command Line Examples

Process raw PIX input from "sample-pix.txt" into triage format in "sample-data.csv:"

```
Parser.pl -c PIX2Triage.parser -i sample-pix.txt -o sample-data.csv
```

Then create a triage view in "sample-data-triage.csv."

```
MessageTriage.pl -i sample-data.csv -o sample-data-triage.csv
```

Or do the same thing in one step (command should be all on one line):

```
Parser.pl -c PIX2Triage.parser -i sample-pix.txt | MessageTriage.pl -o sample-data
-triage.csv
```

Process syslog input from "sample-syslog.txt" into triage format in "sample-data.csv:"

```
Parser.pl -c star.parser -i sample-syslog.txt -o sample-data.csv
```

Then create a a triage view and convert it to a simple HTML page:

```
MessageTriage.pl -i sample-data.csv | csv2html.pl -qbo Triage.html
```

Or do the same thing in one step (command should be all on one line):

```
Parser.pl -c star.parser -i sample-syslog.txt | MessageTriage.pl -i sample-data.csv |
csv2html.pl -qbo Triage.html
```

Excel	Calc
1. Press CTRL-End to go to the bottom of the data. 2. Type any character in the column(s) containing the formula(s). 3. Press CTRL-Home to go to the top of the data. 4. Highlight the formula(s) and choose the Edit, Copy menu (or press CTRL-C). 5. While holding down the shift key, press CTRL-Down Arrow. This should highlight the columns where the formula will go, and it should stop at the characters you typed in step 2. 6. Choose the Edit, Paste menu (or CTRL-V). The new formula should be copied into place.	1. Press CTRL-End to go to the bottom of the data. 2. Type any character in the column(s) containing the formula(s). 3. Press CTRL-Home to go to the top of the data. 4. Highlight the formula(s) and choose the Edit, Copy menu (or press CTRL-C). 5. While holding down the shift key and CTRL keys, press Down Arrow. This should highlight the columns where the formula will go, and it should stop at the characters you typed in step 2. 6. Choose the Edit, Paste menu (or CTRL-V). The new formula should be copied into place.

You may now hide column C. Do not delete it or your formula will lose their source! The may seem like an awful lot of trouble to go to, and it may be. It depends on the circumstances and the other tools you may have available. But the really neat stuff is next. And if your data source can output CSV or other delimited files you'll be able to skip most of the steps above anyway.

Programs

Name	Description
MessageTriage.pl	Take pre-formatted input and create a "triage" view of the data.
Parser.pl	Shell for a generic parser.
FW12triage.parser	A parser.pl configuration file to convert Check Point Firewall 1 logs to the standard input format required by MessageTriage.pl.
Snort2triage.parser	Convert Snort format to triage input.
Syslog2triage	Syslog to triage input.
WatchGuard2triage.parser	WatchGuard Firebox II syslog output triage input.
csv2html.pl	Create a very simple HTML table from CSV input.

Code mentioned in this article is available for download at GoCSI.com.

Notes

❏ Triage works best with discrete messages such as those from IDSs or hosts. Continuous messages such as from firewalls are better analyzed using trending and message thresholds. However, there is still some value to using a triage view. Try it and see if it's useful for you.

❏ I tried to strike a balance between efficiency and readability in the code. Especially in Perl, there's More Than One Way To Do It. Feel free to improve my code. Besides, I am not a real programmer.

❏ Speaking of that last point, this code worked for me and the samples I was able to collect. It could certainly use improvement and may not always work. Your mileage may vary.

Once you've gotten the data into the spreadsheet, the first and most obvious advantage is that you can sort (Data, Sort menu) on different columns. Want to see you data by source host instead of time? No problem. Want to create a chart of messages by time? You can do that too. But there are some other features that can be even more powerful.

The first is AutoFilter (Data, Filter, AutoFilter menu). Autofilter adds pull-down arrows to the top right of each header cell. When you pull down the menu, you get a choice of "All," "Top 10," and "Custom" or Calc's confusingly named "Standard" (both Custom and Standard allow custom criteria). How cool is that? Custom criteria are especially powerful. Want to see every message that contains the phrase "exiting on?" Easy. In Excel, use AutoFilter and a custom filter using the "contains" criteria (scroll down the criteria list). Calc is a little harder; choose a standard filter, choose the [More] button and select "Regular Expressions" with a condition of '=' and a value of ".*exiting on.*" (no quotes).

Subtotals can also be very useful. For example, you can sort by hostname, then find out how many log messages you have from each host. You can also sort messages in one way, apply some kind of color code, then sort another way. It's very interesting what kinds of patterns you can see that way. Another feature was briefly mentioned above. Charting or graphing may allow you to look at some kinds of logs in new and useful ways. There are certainly other possibilities; I've only touched on a few that I use a lot and that I've found most people don't know about.

Is there anything you can do in a spreadsheet that you can't do in other tools? Probably not. Might it sometimes be more trouble than it's worth to use a spreadsheet? Certainly. But you use what's available. And which is more difficult, teaching someone how to use command line tools such as grep, cut, awk, sort and uniq, or sending them the log data they need to review in a spreadsheet that they already know how to use? And you could always use Parser.pl to massage the raw data into CSV format, since that's what MessageTriage.pl needs anyway.

See Also

"A Common Language for Computer Security Incidents" by John D.Howard and Thomas A. Longstaff. - Sandia Report: SAND98-8667,Sandia National Laboratories -http://www.cert.org/research/taxonomy_988667.pdf [32 pages]

JP Vossen, CISSP, is the integration manager for Counterpane Internet Security. He is involved with various open source projects including Snort and he has previously worked as an information security consultant and systems engineer. His two favorite technology toys are Linux and Perl.